VISUAL OBJECT DETECTION AND TRACKING USING LOCAL CONVOLUTIONAL CONTEXT FEATURES AND RECURRENT NEURAL NETWORKS

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

BY

EMRE CAN KAYA

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

SEPTEMBER 2018

Approval of the thesis:

VISUAL OBJECT DETECTION AND TRACKING USING LOCAL CONVOLUTIONAL CONTEXT FEATURES AND RECURRENT NEURAL NETWORKS

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

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

Prof. Dr. Tolga Çiloğlu _____
Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. A. Aydın Alatan _____
Supervisor, **Electrical and Electronics Eng. Dept., METU**

Examining Committee Members:

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

Prof. Dr. A. Aydın Alatan
Electrical and Electronics Eng. Dept., METU _____

Prof. Dr. İlkay Ulusoy
Electrical and Electronics Eng. Dept., METU _____

Assist. Prof. Dr. Elif Vural
Electrical and Electronics Eng. Dept., METU _____

Assoc. Prof. Dr. Nazlı İkizler Cinbiş
Computer Eng. Dept., Hacettepe Universitesi _____

Date: _____

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

Name, Last Name: EMRE CAN KAYA

Signature :

ABSTRACT

VISUAL OBJECT DETECTION AND TRACKING USING LOCAL CONVOLUTIONAL CONTEXT FEATURES AND RECURRENT NEURAL NETWORKS

Kaya, Emre Can
M.S., Department of Electrical and Electronics Engineering
Supervisor : Prof. Dr. A. Aydın Alatan

September 2018, 104 pages

Visual object detection and tracking are two major problems in computer vision which have important real-life application areas. During the last decade, Convolutional Neural Networks (CNNs) have received significant attention and outperformed methods that rely on handcrafted representations in both detection and tracking. On the other hand, Recurrent Neural Networks (RNNs) are commonly preferred for modeling sequential data such as video sequences. A novel convolutional context feature extension is introduced to a proposal-based detection scheme for improving object detection performance. A comprehensive experimental study is conducted to demonstrate the effectiveness of this newly proposed approach. On the tracking side, the effect of several design choices is investigated for an RNN-based tracking algorithm by the help of comparative experiments. Finally, the proposed context feature based method is combined with the RNN-based tracking framework and a joint detection-tracking framework that outperforms the baseline model is proposed.

Keywords: Convolutional Neural Networks, Recurrent Neural Networks, Object Detection, Object Tracking, Context Features

ÖZ

YEREL EVRİŞİMLİ BAĞLAM ÖZNİTELİKLERİ VE YİNELEMELİ SİNİR AĞLARI KULLANARAK GÖRSEL NESNE TESPİTİ VE TAKİBİ

Kaya, Emre Can
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. A. Aydın Alatan

Eylül 2018 , 104 sayfa

Görsel nesne tespiti ve takibi; gerçek yaşamda bir çok önemli uygulama alanı bulunan, bilgisayarlı görünün iki başlıca problemidir. Geçtiğimiz onyılda, ayırt edici olarak eğitilen görsel temsil modelleri olan Evrişimli Sinir Ağları'na (ESA) ilgi arttı ve bunların performansı el yapımı temsillere dayanan modelleri hem tespit hem takip probleminde geride bıraktı. Öte yandan, videolar gibi sıralı yapıya sahip verileri modellemede Yinelemeli Sinir Ağları (YSA) sıkça tercih edilmektedir. Hipotez temelli tespit yöntemlerinin performansını arttırmaya yönelik olarak yenilikçi bir evrişimli bağlam özniteliği eklentisi önerilmektedir. Önerilen yeni yaklaşımın etkililiğini göstermek adına, geniş kapsamlı bir deneysel çalışma yürütüldü. Nesne takibi tarafında, karşılaştırmalı deneyler yoluyla çeşitli tasarımsal seçimlerin YSA temelli bir takip algoritması üzerindeki etkileri incelendi. Son olarak, önerilen bağlam öznitelikleri temelli yöntem YSA temelli takip algoritmasıyla birleştirildi ve birleşik tespit-takip algoritmasının tespitsiz algoritmaya göre daha iyi bir takip performansı sergilediği gözlemlendi.

Anahtar Kelimeler: Evrişimli Sinir Ağları, Yinelemeli Sinir Ağları, Nesne Tespiti, Nesne Takibi, Bağlam Öznitelikleri

To my family...

ACKNOWLEDGMENTS

First of all, I would like to present my gratitude to Prof. Dr. A. Aydın Alatan for his valuable guidance and constant support in this study. Under his supervision, sky seemed to be the only possible limit. Also, I am intensely grateful to Dr. Alper Koz who taught me how to approach a research problem in a systematic way and has been always there to help whenever I asked for. I am strongly appreciative of all the brilliant and inspiring ideas, suggestions and solutions provided by Yeti Ziya Gürbüz, Mustafa Ergül, Dr. Erhan Gündoğdu, Oğul Can, Dr. Gökhan Koray Gültekin and Mertalp Öcal. I always felt so lucky for having the chance to work together with such intelligent and awesome people.

I am also deeply grateful to ASELSAN Research Center for supporting this study.

Furthermore, I would like to respectfully thank to all the wonderful people I met in OGAM including Mustafa Kütük, Kutan Feyiz, Esat Kalfaoğlu, Akın Çalışkan, Sefa Mert Tarhan, Onur Barut, Ece Selin Böncü, Aybüke Erol, Aziz Berkay Yeşilyurt, İhsan Emre Üstün, İzlen Geneci, Hilmi Kumdakçı, Havva Oğuz and Beril Beşbınar. Without their friendship and companionship, I wouldn't be able to make it till the end.

Finally, I would like to thank to my family by their spiritual support during my thesis study.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
SVM	Support Vector Machine
HOG	Histograms of Oriented Gradients
RoI	Region of Interest
IoU	Intersection over Union (Jaccard Overlap)
VOT	Visual Object Tracking
EAO	Expected Average Overlap
AP	Average Precision
mAP	mean Average Precision
ReLU	Rectified Linear Unit
NMS	Non-Maximum Suppression
BPTT	Backpropagation Through Time

CHAPTER 1

INTRODUCTION

Computer vision is an interdisciplinary field of research that aims at developing algorithms to extract useful information from visual data. Subjective opinions of humans is often decisive in determining the useful information. It is often desired that the algorithm responds to an input in a similar way to how human visual system does when developing computer vision algorithms. When somebody examines a photo of a cat, one usually assumes that most of the other humans would agree that the photo contains a cat. On the other hand, there is not a clear mathematical formula to find out whether there is a cat in the photo or not. In certain cases, it may be unclear whether the photo belongs to cat or a dog, so that not all humans give the same response when it is asked. Visual recognition is in this sense subjective. Computers, on the other hand, carry out mathematical operations that are strictly based on a set of algebraic rules. This may, or may not, be similar to the way the human mind works. In fact, it is not clearly known how human mind makes decisions. Although the internal behavior of an algorithm may be completely different than that of human visual system, an algorithm is considered a successful one if its input-output relationship is close to human vision.

Still, as in many other technical fields, living things are a major source of inspiration in computer vision and recent research favors biologically plausible models. Convolutional neural networks (CNNs) are such models that bear resemblance to their biological counterparts. Their multi layer structure is reminiscent of the stratified nature of human visual cortex.

Convolution operation is commonly used in many fields of engineering to describe

the behaviour of Linear Time Invariant (LTI) systems. Convolution allows us to easily compute the output of an LTI system to an arbitrary input by accumulating individual responses to impulses that are thought to constitute the arbitrary input. In CNNs, learned filters can be thought of as impulse responses that are sensitive to certain visual features. In this context, time invariance is replaced by *shift invariance*, which suggests that when a visual entity is shifted by a certain amount in the input, the corresponding response is the same but shifted by the proper amount in the output.

On the other hand, a closely related area of research to computer vision is machine learning. Machine learning aims to build algorithms to make machines learn. Machine learning became essential for computer vision in the recent years. Learning is usually formulated as a problem of optimization. There is, however, an important difference between a conventional optimization problem and a machine learning problem: Convential optimization seeks to find the best solution using all of the available data. The solution is relevant only to the data being used. Therefore one cannot use the solution for similar data.

In order to exemplify this, consider the task of planning a travel route from Ankara to Kars. One might define certain criteria for a desired route: A quiet long route is not desirable. A route that goes through regions with poor road conditions is also not desirable. While going all the way, one might want to stop by in nearby places with tourist attractions which should count as desirable changes to the route. By quantifying all such considerations with the appropriate mathematical formulation, one may construct an optimization problem that seeks the best route from Ankara to Kars. The resulting solution is applicable to Ankara-Kars trip only. If next year, somebody decides to travel to İzmir instead, then one would probably apply the same procedure without using any computed result from the first trip.

Instead of explicitly defining criteria for the best route, imagine one could ask a lot of people about their trips around the world and which routes they enjoyed the best. One can collect data which are relevant to the problem such as tourist attractions, road conditions, road lengths and so on. Then one can construct a model with some unknown parameters that will decide how the response should be to a certain input. Unknown parameters can be learned from collected data and responses from people

corresponding to this data. Learning is achieved by optimizing the response of the model. Such a model falls into the range of machine learning and is to be called a learned model.

In machine learning, data is essential for training a model. By the year 2017, number of digital photos taken worldwide was estimated to be 1.2 trillion which is about twice the number estimated for the year 2013 [27]. This overwhelming increase in digital photography is largely due to widespread usage of smartphones in the recent years. Smartphones not only facilitate photography, they also boost the sharing of visual data via social media. About one fifth of the digital photos taken are shared on social media platforms, such as Facebook and Instagram [28]. Social media make visual data publicly available. Availability of large amounts of visual data has paved the way to the domination of algorithms that learn representations from data itself in almost every field of computer vision. The main advantage of learned representations against handcrafted representations is that one can improve the representation by simply feeding additional data.

Objects are thought to have discriminative features that are helpful when recognizing them. Features are easy to define when spoken in words. For example, a German Shepherd dog has upright ears that clearly distinguishes it from a Saint Bernard or a Cocker Spaniel. Unfortunately, computers do not speak in words, they compute in ones and zeros. Goal of representation learning is to mathematically formulate discriminative features so that they can be computed.

Beside the object features that help to discriminate between different objects, visual context is also shown to play a significant role in human vision. According to numerous experimental studies [29, 30, 31, 32], when recognizing objects, our visual systems take into account certain contextual cues that establish a relation between an object and its surroundings. Motivated by these findings, object recognition algorithms that aim to model and exploit context have been developed [33, 34, 16, 35, 36, 37, 38].

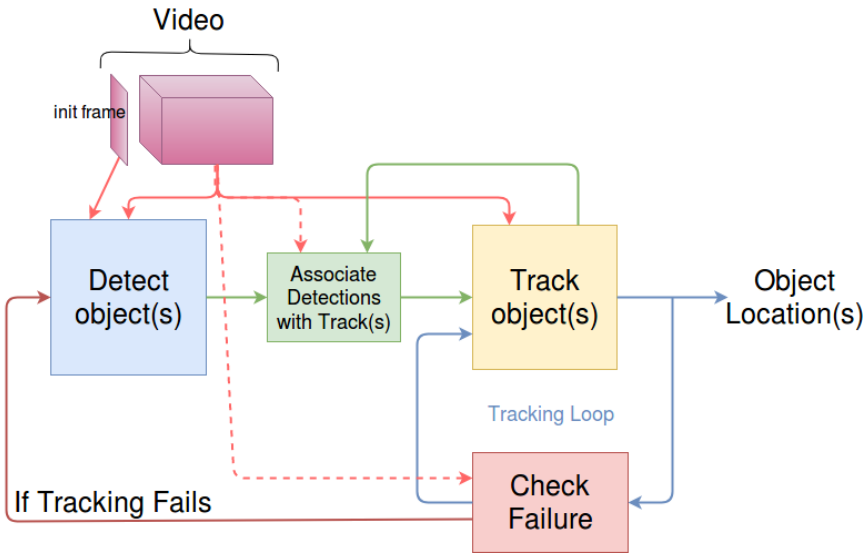

Figure 1.1: A model that combines tracking and detection.

1.1 Problem Statement and Motivation

Object Detection and Tracking are two important problems of computer vision. While they are two distinct problems, in most real life scenarios, they often occur in conjunction and they are strongly related. Object tracking has applications in a wide range of areas including medical diagnosis systems, visual surveillance and security systems, video games, industrial robotics and traffic monitoring [39]. In almost all of these applications, a prior detection step is necessary to determine the initial location of the object(s) to be tracked. A typical real-life scenario that incorporates object detection and tracking is depicted on Figure 1.1. In this scenario, objects of interest are determined through detection on the initial frame of the video stream and a track is initiated for each object. Then the system enters a tracking loop in which the most recent estimation for the object location is fed back to the tracker. Tracking loop can be broken for one of the tracks if the system detects a failure. In this case, detection is performed on the current frame to obtain possible locations for the objects of interest. Regardless of whether it is a multiple or single target tracking scenario, data association needs to be performed to associate object detections to track(s). After this, the system reenters the tracking loop.

Visual object detection is the problem of estimating the class and location of objects in visual data. In many real-life applications one is often interested in two forms of

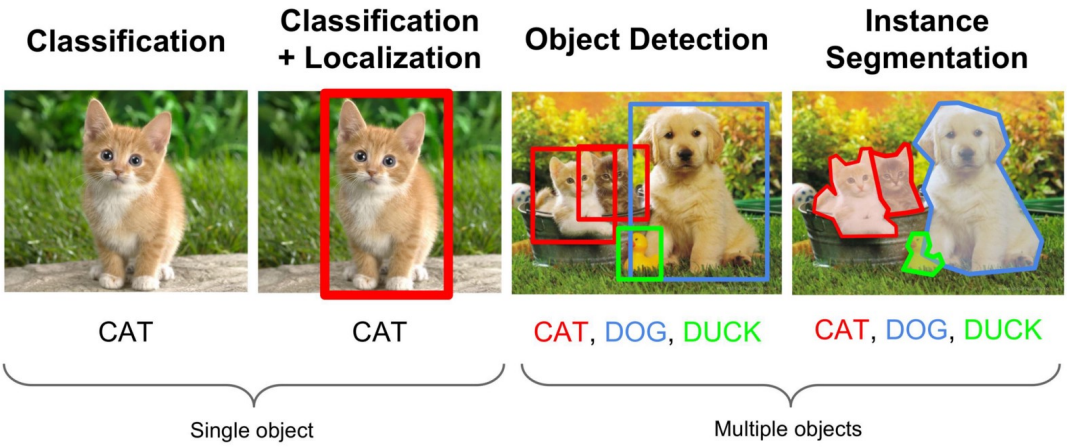

Figure 1.2: Image classification, object detection and instance segmentation. Image taken from: [1]

visual data: Photographs and videos.

Figure 1.2 compares the problem of object detection to three closely related problems. Image classification can be applied to determine the class of the object when it is assumed that a single object dominates an entire image. In certain applications, there is a single object of interest in an image and its location and class is not known. In this case, classification and localization are the two problems to be solved for that object. In object detection, aim is to classify and localize all the objects in an image that belong to a set of predefined object classes. In instance level segmentation, objects are localized by their full extent rather than bounding boxes that roughly describe the location.

In ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [40], object detection is defined as the task of producing a list of object categories present in the image along with an axis aligned bounding box indicating the position and scale of every instance of each object category.

Visual object tracking deals with moving objects in videos. In visual object tracking, the aim is to estimate the location of an object in all frames when the initial location is known. Visual object tracking has various application fields including medical diagnosis, industrial robotics, traffic monitoring, surveillance and security systems. It should be noted that in most of these applications tracking is preceded by object

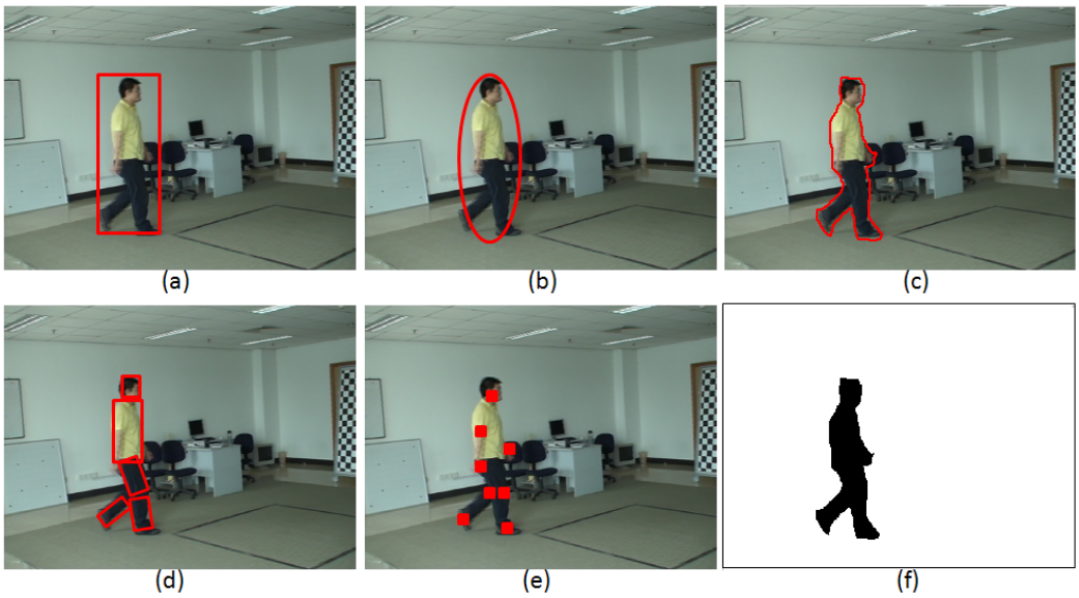

Figure 1.3: Forms of object localization in tracking: (a) bounding box, (b) ellipse, (c) contour, (d) articulation block, (e) interest point, (f) silhouette. Image taken from: [2]

detection. Initial location of tracked objects are usually provided by a detection algorithm and this step is called automatic initialization of objects.

A video is a sequence of frames that are ordered with respect to corresponding time instants they capture. Motion is perceived, if the frames are viewed consecutively in an appropriate speed while watching a video. From this perspective, a video is not merely a collection of images. Consecutive frames are similar to each other up to some degree so that human brains perceive moving objects. Frames contain common information that is necessary for the task of tracking an object. In visual object tracking the aim is to efficiently extract and utilize this useful information for locating the object in consecutive frames. In tracking, object localization can take various forms as depicted on Figure 1.3. Among these forms, we are mostly interested in bounding box tracking.

Objects in videos are subject to several types of appearance changes which pose challenges when tracking them. Different types of appearance changes are exemplified on Figure 1.4.

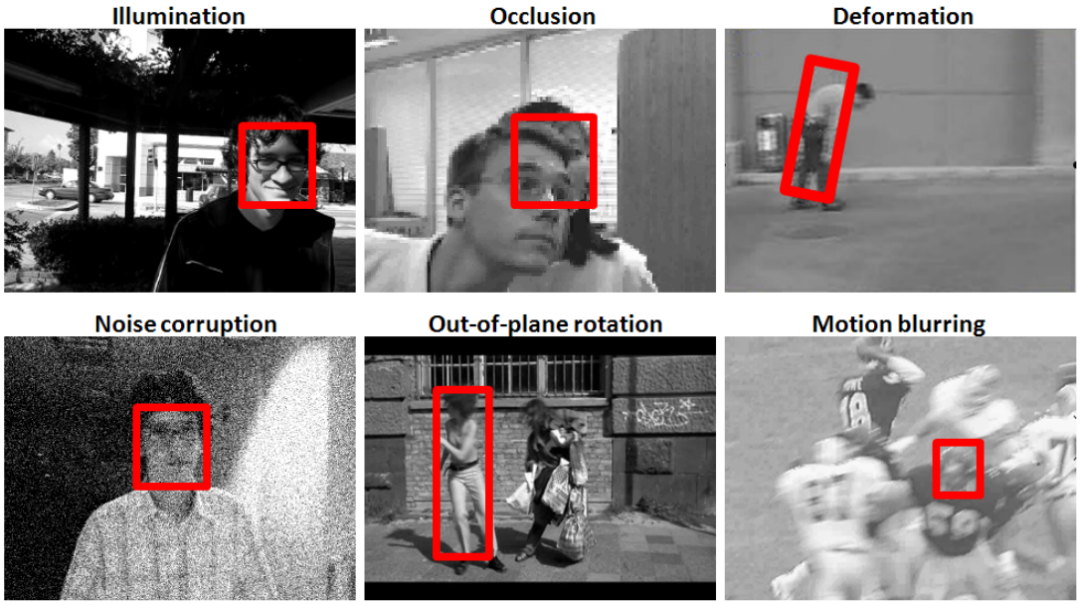

Figure 1.4: Types of apperance changes in visual objects. Image taken from: [2]

1.2 Scope of the Thesis

This thesis has three main contributions: A proposal based object detection scheme that uses local convolutional context features to improve detection performance is developed. A comprehensive comparative study to investigate visual object tracking using recurrent neural networks is conducted. Finally, an object tracking framework that combines context feature based detection and recurrent neural network based tracking is developed.

1.3 Outline

- In Chapter 2, we provide a comprehensive review of the recent developments in the field of object detection.
- In Chapter 3, we present our context feature based extension to proposal-based object detection.
- In Chapter 4, we provide a comprehensive review of the recent developments in the field of visual object tracking.
- In Chapter 5, we present the Re^3 algorithm [41] and extensive experimental analy-

sis is carried out to investigate the operation of Re^3. Furthermore, we develop a detection-supervised tracking framework which combines Re^3 [41] and our context feature based object detection model.

CHAPTER 2

VISUAL OBJECT DETECTION LITERATURE

In visual object detection, the goal is to predict a bounding box and a class for each object in an image that belongs to a set of predefined classes. Output bounding box should cover the full visible extent of the object. A naive approach to this problem is to slide windows at various scale and aspect ratios over the whole image, classify the window contents and return the windows that are most likely to contain objects as a detection result. Sliding window detection is a brute-force approach that degrades the problem into a set of classification problems. The main disadvantage of sliding window approach is its computational inefficiency. A more recent paradigm of object proposal generation suggests to reduce the set of possible object windows with much less computation prior to performing classification. Apart from that there are also methods that perform one-shot detection with no sliding window or proposals.

In object detection, classification, segmentation and localization; transforming raw visual data into semantically meaningful forms plays an important role. The outputs of such transformations are usually called *features*. Features are useful, if they are discriminative so that they contribute distinguishing different objects. In conventional approaches, features are obtained by fixed, data independent tranformations that rely on expert knowledge. Resulting features are called *manually crafted* or *hand-crafted features*. On a different track, learned representations are used in which, the features are learned from the data itself. In learning representations, the amount of training data is decisive on the performance of the algorithm.

It should also be noted that a frequently applied post-processing technique in object detection is non-maxima suppression (NMS). Object detectors usually return a lot

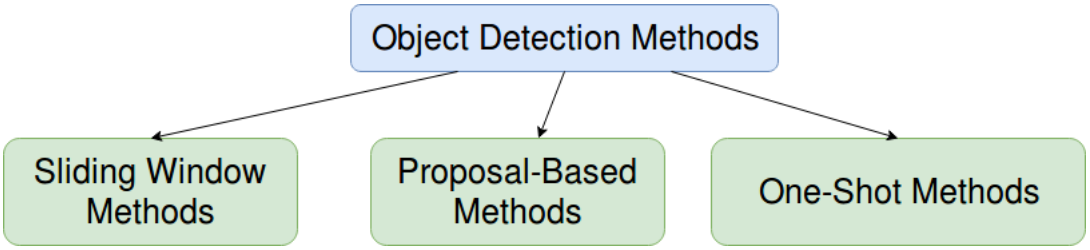

Figure 2.1: Taxonomy of Object Detection Methods.

of bounding box predictions that are likely to capture the same object more than once. NMS eliminates the predictions that have high overlap with more confident predictions. NMS is usually applied at test time however there are also cases that it is applied during training to reduce computation.

According to the ground truth content they utilize, object detection methods can be fully-supervised (bounding box annotations are available), weakly-supervised (only image-level annotations are available) or unsupervised. Our focus is mainly on the fully-supervised methods.

Fully-supervised object detection methods can be grouped into three classes as sliding window based methods, methods that are built on top of proposal generators (see Sec. 2.2.1) and methods that do not generate any proposals called one-shot detection methods.

2.1 Sliding Window Object Detection

In the past, object detection algorithms were built on manually crafted features (descriptors) such as Histograms of Oriented Gradients (HOG) [42], Scale-Invariant Feature Transform (SIFT) [43] and Local Binary Patterns (LBP). In sliding window detection; windows with various size, location and aspect ratio are hypothesized without using any image information other than the height and width of the image. Patch features extracted using these windows are classified to decide whether it contains an object of a known category or not. Classification should be quite fast in such a method to scan the whole image in an acceptable time. Nowadays, the usage of handcrafted features are much more limited due to the success of learned representations.

Moreover, sliding window approach is replaced by more time-effective approaches.

HOGs are constructed by computing intensity gradients among a finite set of directions in a dense grid and expressing the gradient distribution over this finite set of directions. Dalal and Triggs [42] used HOG descriptors in a sliding window fashion to detect humans. A linear SVM is trained for this binary classification problem.

More recently, Deformable Parts Model (DPM) [44] was introduced. In DPM, a feature pyramid of HOGs is constructed for an input image. Each level of the pyramid corresponds to a set of features in different resolutions. A deformable parts model consists of several components that correspond to different views of objects and each component consists of a root filter and corresponding part filters that capture the features of different object parts. An object consists of parts and parts can be in different spatial configurations in an image. These different configurations are modeled by the positions of part filters with respect to a root filter that centers the object. In DPM, object parts are modeled as rectangular regions. When a part deviates from its ideal location, it is called a *deformation*. The model is learned through a latent SVM formulation.

2.2 Object Proposal Generation and Proposal-Based Object Detection

Many detection algorithms are built on object proposals which are generated by a separate object proposal method. In this case, detection performance is usually upper bounded by the correct detection ratio (i.e. recall) of the proposal method. In this section, we shall first provide a brief survey of the most prominent object proposal methods.

2.2.1 Object Proposal Methods

Object proposal generation problem is formulated as follows: Given an input image, the task is to return a set of bounding boxes or regions that are likely to contain objects with a corresponding set of scores measuring the likelihood of containing an object.

High recall and efficiency are the main concerns when designing object proposal al-

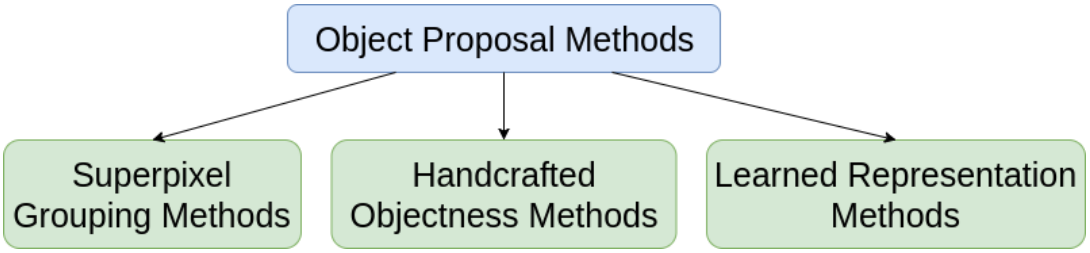

Figure 2.2: Taxonomy of Object Proposal Methods.

gorithms. An object that is undetected (i.e. missed) in the proposal stage, is not likely to be recovered in the later stages of detection. Computational efficiency is also a key aspect in proposal generation. It should be noted that the aim in the first place in using a proposal method is to reduce the computational cost. Object proposal generation can be seen as a stage that casts the detection problem into a classification problem. On the other hand, some detection methods involve a bounding box regression stage as a post-processing step (after obtaining initial detections) or as a jointly trained module. In that case, the bounding box of the final detection is not necessarily contained within the set of proposals. Hence, the detector part of the framework does more than classifying and ranking a bounding box. It can modify the bounding box.

Object proposals come in two forms: A bounding box or an arbitrary shaped segment. Bounding box annotations are easy to produce and they provide sufficient location information for many real-life applications.

Based on the underlying philosophy, object proposal methods can be classified into the following sub-classes: superpixel grouping methods, handcrafted objectness methods and learned representation methods, as depicted on Figure 2.2. It is important to note that proposal methods can be categorized in several other ways. Given categorization is useful as far as it helps to develop an understanding on well-known proposal methods.

2.2.1.1 Superpixel Grouping Methods for Object Proposal Generation

Superpixel grouping methods usually initialize with an oversegmentation step that divides an input image into its superpixels. Superpixels are defined as small regions of uniform color and brightness. These small regions are usually contained within a

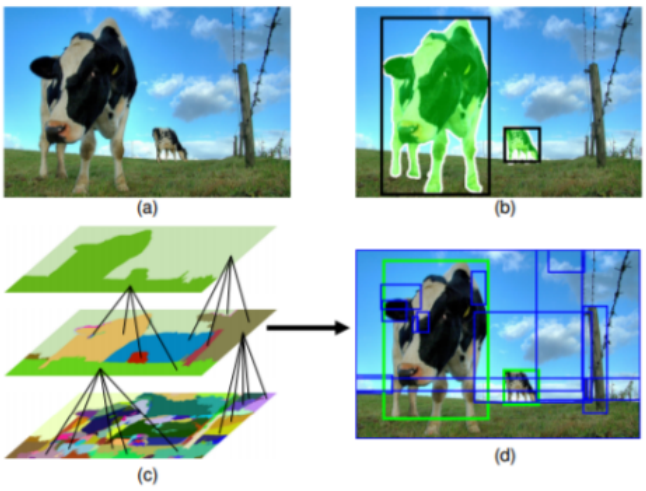

Figure 2.3: Selective Search aims to generate object proposals by hierarchically merging similar neighbour regions. (a) Input image, (b) Ground truths, (c) Hierarchical Grouping of regions, (d) Bounding Box Proposals generated by Selective Search. [3].

single object. After obtaining superpixels, larger regions are obtained by growing or merging superpixels. Notable segmentation based proposal methods include Selective Search [3], Geodesic [6], Randomize Prim's [45], Constrained Parametric Min-Cuts [5] and Multiscale Combinatorial Grouping [4].

Selective Search initally applies the graph-based segmentation method proposed by Felzenszwalb and Huttenlocher [46] to obtain superpixels. According to certain predefined similarities, neighbor regions are iteratively merged into larger regions. Similarity measures involve color histograms, low level features, size and shape information. Selective Search is visualized on Fig. 2.3.

Multiscale Combinatorial Grouping (MCG) [4], follows a top-down procedure, where initial segmentations of varying size and resolution are obtained from the different scales of the same image via normalized cuts. Boundaries generated at individual scales are used to train a boundary classifier which produces boundary estimates at single-scale. This step corresponds to the "Combination" block shown on Fig. 2.5. Combinatorial grouping is applied to obtain object candidates out of boundaries. In this step, number of possible regions is reduced to an acceptable range by formulating the problem as a Pareto front optimization. The remaining regions are then ranked by

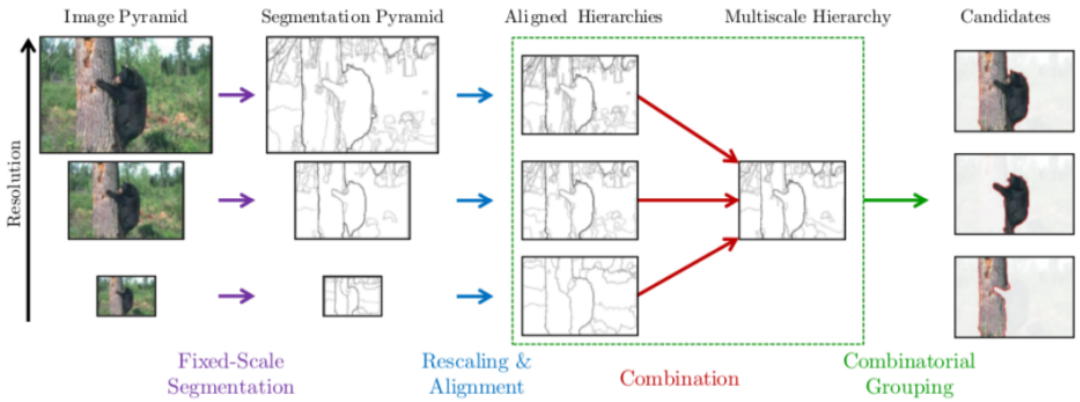

Figure 2.4: Multiscale Combinatorial Grouping [4].

a random forest according to low level features, shape, size and contour information. MCG can be considered a segmentation based method. The method is summarized on Fig. 2.5.

Constrained Parametric Min-Cuts (CPMC) [5] starts by placing foreground seeds on a regular grid on the image and background seeds along the borders. For each foreground seed region, an independent binary min-cut problem [47] to segment foreground and background is solved. The energy function applied to solve the min-cut problem consists of unary and pairwise potentials calculated for each pixel. Unary potential involves a variable foreground bias term so that by varying bias one can obtain several proposal regions for a foreground seed. These proposals are finally ranked by applying linear regression or random forests, similar to the ranking step in MCG. Note that this final step involves supervised learning from ground truth object regions.

Geodesic Object Proposals [6] starts with an oversegmentation as in Selective Search and MCG. Based on this initial segmentation, the algorithm places seeds similarly to CPMC. Geodesic seeds are placed by classifiers specifically designed for discovering objects which is different from the case in CPMC. Oversegmentation can be represented as a graph together with the corresponding boundary probability map where nodes are the superpixels and edge weights are boundary probabilities. Using edge weights, one can compute a distance between the nodes of a graph. Geodesic distance between the two nodes is defined to be the length of the shortest path between them. Geodesic distances between the seeds are used as seed features together with

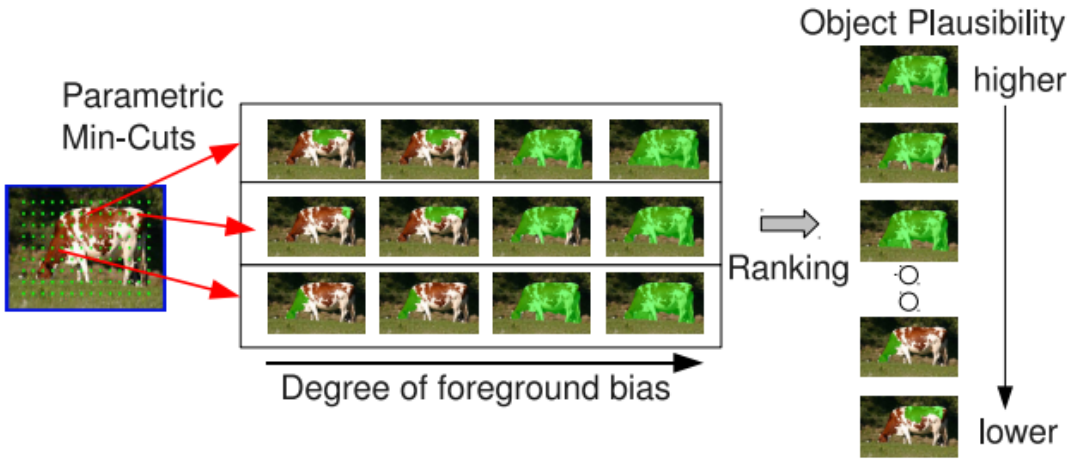

Figure 2.5: Constrained Parametric Min-Cuts [5].

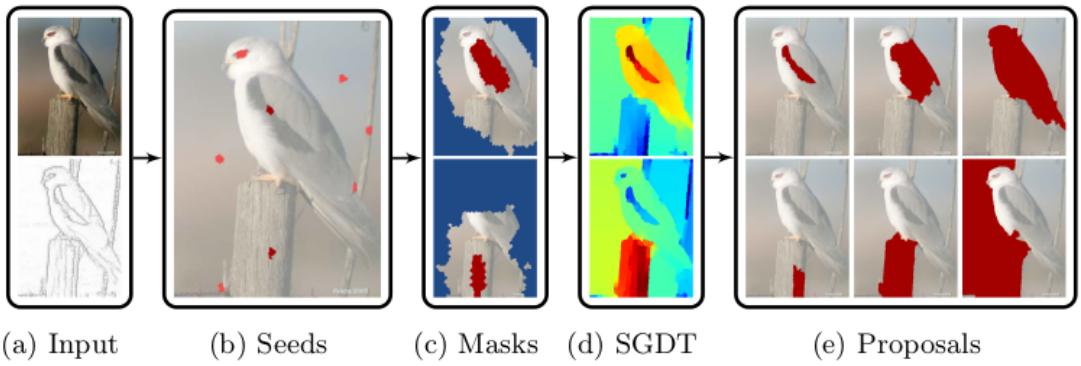

Figure 2.6: Geodesic Object Proposal Generation Method: (a) Input Image and the corresponding oversegmentation. (b) Seed placement. (c) Foreground-background masks obtained from individual seeds. (d) SGDT maps. (e) Final results. [6].

other cues, such as color and position. Seed placement is governed by a classifier acting upon these features. After the seeds are placed, a foreground-background classifier (which is trained on object ground truths), generates masks. According to these masks, each superpixel is assigned a Signed Geodesic Distance Transform (SGDT) value calculated from geodesic distances. SGDT measures the likelihood of the pixel of belonging to foreground or background. Sharp changes in SGDT map are used to identify object proposals. The algorithm is schematized in Fig. 2.6.

2.2.1.2 Handcrafted Objectness Methods for Object Proposal Generation

Apart from the superpixel based approach discussed above, there is a set of techniques for proposal generation that are built upon the "objectness" concept, namely *objectness-based methods*. Objectness, coined by Alexe et al. [48], is the property of a bounding box. Objectness expresses how likely it is that the box corresponds to an object. Objectness can be a property that is learned by a CNN [49] or a property that is defined solely over the edge maps [50]. When compared to superpixel grouping methods, objectness-based methods have the obvious difference that they do not actually eliminate the need for sliding window. Initially, sliding window needs to be applied to generate an initial set of windows over which objectness measure can be applied. Therefore objectness-based methods should be quite fast so that they have an advantage over applying sliding window detection directly. Although objectness can be defined through learned representations as well as handcrafted representations, in this section the objectness methods are gathered that are based on handcrafted representations. Section 2.2.1.3 is devoted to learned representation methods.

Alexe et al. [48] derive objectness from *image cues*: Multi-scale Saliency (MS), Color Contrast (CC), Edge Density (ED) and Superpixels Straddling (SS). Saliency has its roots in cognitive neuroscience as a fundamental concept for understanding visual perception. Multi-scale saliency cue relies on a spectral residual based saliency definition [51]. CC cue is based on the assumption that an object should exhibit a colorwise dissimilarity to its immediate surroundings. ED captures the contour information. SS cue quantifies how much the bounding box borders pass through the superpixels. This algorithm uses the same superpixel segmentation algorithm applied in Selective Search [3]. Each of these cues aims to capture a different aspect of objects. A single cue serves as a necessary but not sufficient criterion for being an object. Alexe et al. [48] propose a supervised Bayesian scheme for learning the image cue parameters.

EdgeBoxes [50], a more recent method, identifies objects from their edges by estimating the number of contours contained in a box. In that approach, objectness is solely defined over edges. Contour Box [52] uses closed path integrals on contours and degrades the problem into an optimization of completeness and tightness of contours

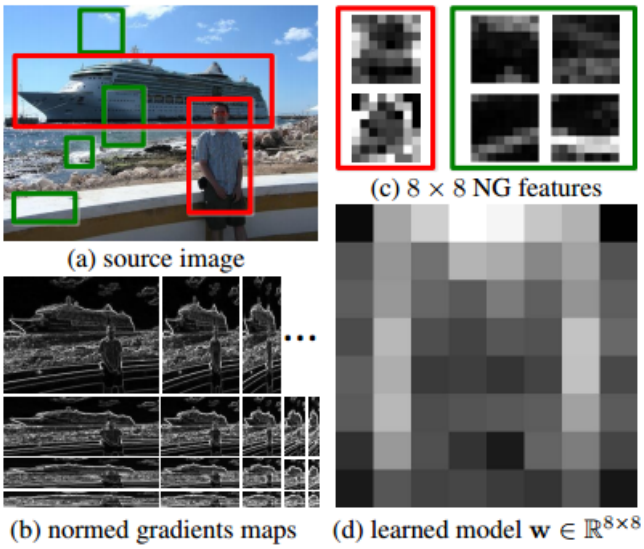

Figure 2.7: Objectness estimation with Binarized Normed Gradients [7].

which can be efficiently solved in polar coordinates.

Cheng et al. [7] propose a much more simpler and efficient objectness measure which they call Binarized Normed Gradients (BING). BING extracts a fixed size binarized normed gradient map as the only descriptive feature of each bounding box and rank these features with a learned linear model as summarized in Fig. 2.7.

2.2.1.3 Learned Representation Methods for Object Proposal Generation

As in many machine vision tasks, CNNs proved to be successful in object detection and object proposal generation as well. CNNs provide powerful learned visual representations. The main advantage of using learned representations is that they can be improved indefinitely by simply adding more training data. Here we shall describe some of the well-known CNN-based proposal methods. These methods can be further divided into two as methods that utilize off-the-shelf CNN features and methods that learn representations for the task of proposal generation.

One such method is DeepProposal [8], which makes use of the representation ability of CNN features learned from large amount of data instead of making strong assumptions. These features are used off-the-shelf. Edges and object contours that are frequently utilized in proposal generation also plays an important role in this method.

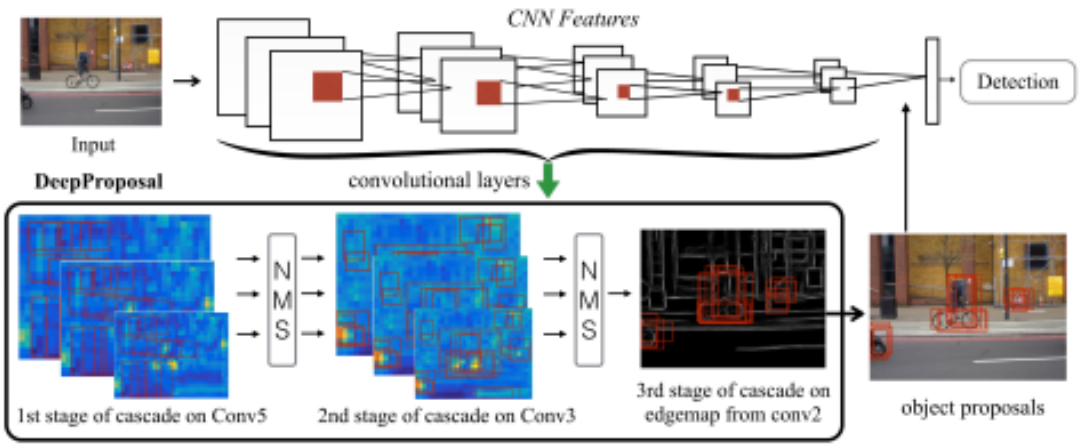

Figure 2.8: DeepProposal object proposal framework. [8].

In DeepProposal, object contour detection is learned from CNN features by structured random forests [8] which is a radically different approach from the previously mentioned methods. DeepProposal is summarized in Fig. 2.8.

DeepMask [9] is a fully convolutional model that generates object proposal regions in the form of arbitrary shaped segments. In this method, CNN features are transferred from an Imagenet [40] pretrained model and finetuned on the task of object proposal generation. The network is trained end-to-end with a joint loss that quantifies the correspondence of output segmentation with the ground truth object mask and output object score with the binary label (object or not). DeepMask network structure is depicted on Fig. 2.9. SharpMask [53] is an improved version of DeepMask, where features from different layers of the CNN are blended in a bottom-up/tow-down network architecture to generate sharp object masks. The main drawback of DeepMask is that it can return only a single proposal as a result of a single network evaluation on an image patch. For obtaining multiple proposals, the image should be scanned by the network in a sliding window fashion. Computational cost grows proportionally with the number of objects to be searched. A method similar to DeepMask is InstanceFCN [54] which generates instance-sensitive feature maps in a fully-convolutional setting. InstanceFCN can handle several object instances in a single forward pass.

On the other hand, Jie et al. [10] employ a fully-convolutional network to predict bounding box proposals instead of object masks. In this approach, a confidence map is generated where each pixel value serves as a confidence value for the proposal box

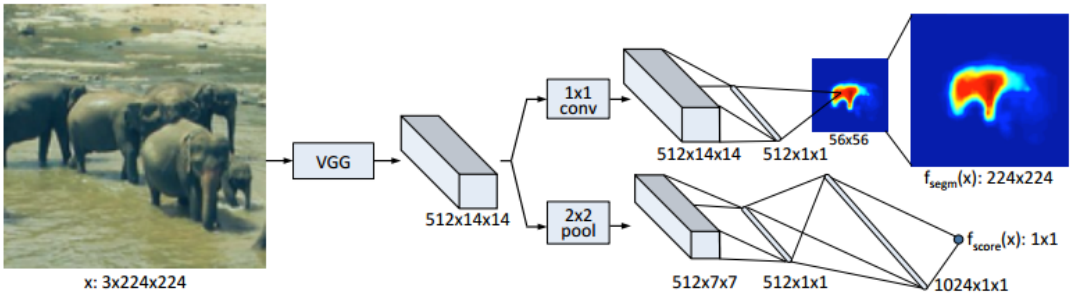

Figure 2.9: DeepMask network architecture for object candidate segmentation [9].

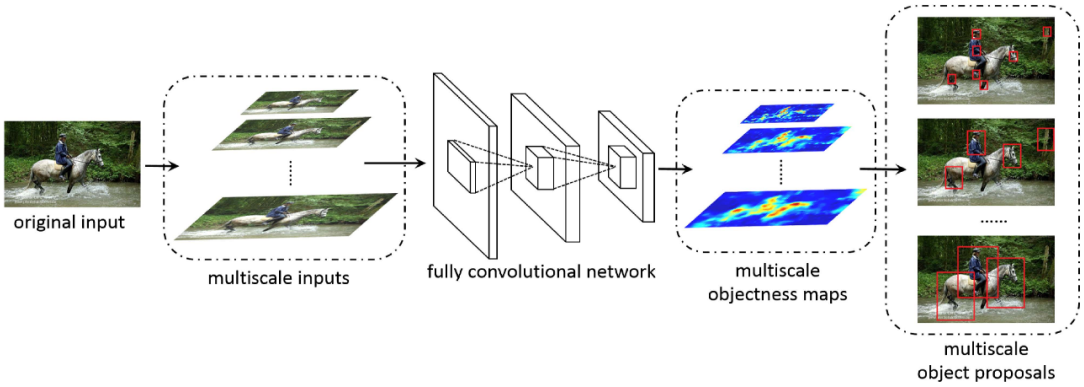

Figure 2.10: Object proposal generation with a fully convolutional network [10].

derived from the receptive field of the pixel. The main drawback of this method is that the input image needs to be fed multiple times in different scales and aspect ratios to account for varying scale and aspect ratio proposals. The method is depicted on Figure 2.10.

In DeepBox [49], a small size convolutional neural network is trained to classify boxes as object or not object. Proposals generated by a bottom-up method such as Selective Search are reranked according to their CNN features (a top-down approach), to obtain better proposals. Multibox [55] [56] on the other hand, formulates object bounding box prediction as a regression task. In this method, an end-to-end trained CNN predicts a fixed number of bounding box coordinates with the corresponding confidence values.

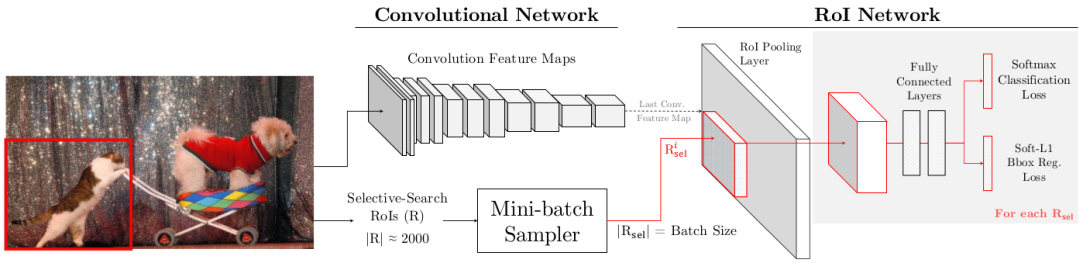

Figure 2.11: Fast R-CNN [11] (Figure taken from: [12]).

2.2.2 Proposal-Based Object Detection

Development of many successful object proposal methods has paved the way to the rise of proposal based detection. Shortly after this, excellent representational capabilities of CNNs were demonstrated on large-scale image classification by Krizhevsky et al. [57].

Motivated by the success of CNNs on image classification, R-CNN [58] was introduced as an object detection framework based on CNN features. R-CNN is a multi-stage feedforward model that consists of region proposal generation (via Selective Search [3]), CNN feature extraction, classification (multiple SVMs) and bounding box regression stages. R-CNN uses CNN as an off-the-shelf feature extractor.

Fast R-CNN [11] is an improved version of this algorithm. In Fast R-CNN, Region of Interest (RoI) pooling layer is introduced. RoI Pooling layer pools variable shape RoI (proposal) features into fixed size features. Detection speed is improved by computing the convolutional features for the whole image instead of for each proposal separately. SVM classifiers and separate bounding box regression model are replaced by neural network classifer and regression layers that can be trained in conjunction with the rest of the network. Error can be backpropagated through the RoI pooling layer which allows for finetuning of the visual features that were transferred from a classification network for the detection task. Fast R-CNN is depicted on Fig. 2.11. More recently, Faster R-CNN [17] was introduced. The essential difference from Fast R-CNN is that the CNN itself generates its own proposals via convolutional layers instead of relying on external region proposals. This allows the model to be end-to-end trainable.

Dai et al. propose R-FCN [13] architecture that employs the *position-sensitive RoI*

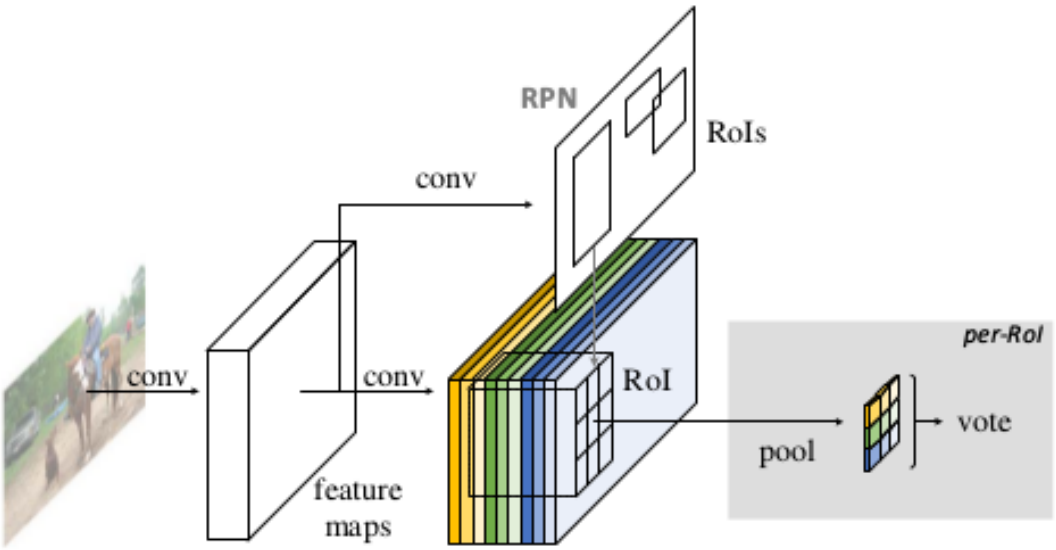

Figure 2.12: Region-based Fully Convolutional Networks (R-FCN) [13].

pooling layer which pools scores from different score maps which are associated with different subregions in a RoI. A RoI is divided into cells by a regular grid to define the subregions. They apply voting over these subregion scores to obtain the final classification for a region of interest as depicted on Fig. 2.12. Obtaining score maps for the whole image and applying voting over these maps eliminates the need for a fully connected classifier and computational cost is significantly reduced.

In DeepID-Net [59], deformable part based modelling was incorporated into R-CNN framework. DeepID-Net employs the so-called *def-pooling layers* in between convolutional layers. Def-pooling layers perform max pooling according to learned deformation constraints.

Hypernet [60] extends the architecture in Faster R-CNN by combining the feature maps from multiple scales to obtain the so-called *hyper feature maps* to account for multiple resolutions. A similar work is Feature Pyramid Networks (FPN) [61]. FPN uses a bottom-up/tow-down architecture with skip connections similar to Sharpmask [53] to make bounding-box predictions at multiple scales. Both Hypernet and FPN aim at improving detection performance of Faster R-CNN by efficiently combining the information from multiple convolutional layers instead of using the final layer output only. Information from earlier layers is important for especially recognizing small objects.

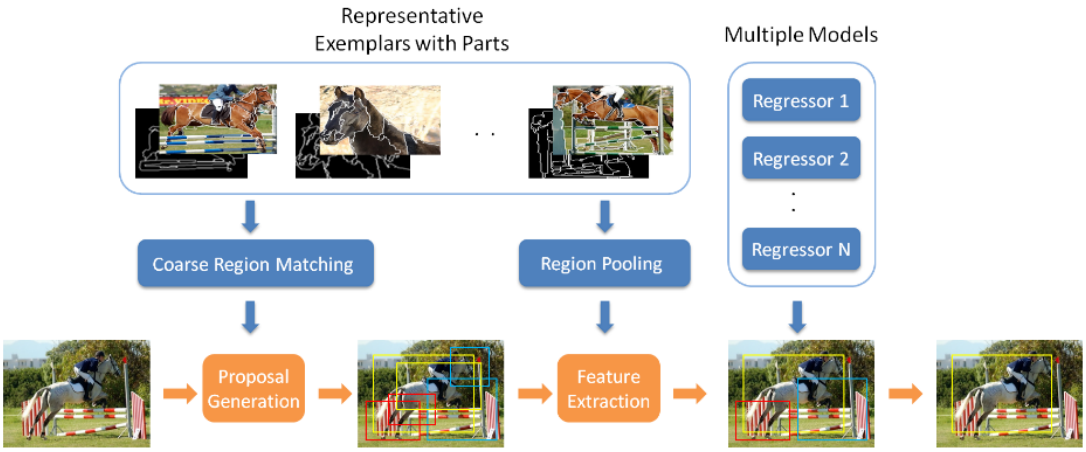

Figure 2.13: Object Detection with Adaptive Region Pooling [14].

Tsai et al. [14] propose a part-based detection framework called Adaptive Region Pooling in which, parts are defined to be arbitrary shaped, non-rigid regions that together form the object. In this framework, for each object class a set of representative exemplars are discovered from the training set. An externally generated object proposal is matched to an examplar according to their similarity in shape and appearance. After this step, part features of the target object are extracted according to the part segmentation of the exemplar. Multiple Support Vector Regressors (SVR) are applied to pooled part features to generate final detection results. Adaptive Region Pooling is depicted on Fig. 2.13.

2.3 Single-shot Object Detection

In one-shot detection methods, detection results are obtained from a single network evaluation for the whole image and they are simply proposal-free. Computational complexity is significantly reduced by the elimination of proposal generation. Prominent representatives of this group of methods are YOLO (You Only Look Once) [15, 62] and SSD (Single shot multibox detector) [63]. In YOLO, an input image is divided by a regular grid where the resulting image patches (grid cells) are assigned a set of class probabilities. For each grid cell, a number of bounding boxes are predicted by the network. A grid cell is responsible for detecting an object if the center of an object falls into it. This procedure is visualized on Fig. 2.14. Although

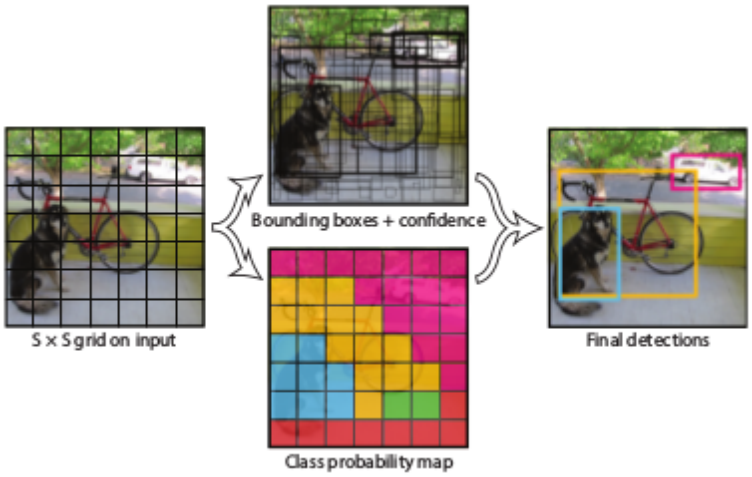

Figure 2.14: YOLO divides an input image into a regular grid of cells to predict bounding boxes and class for each cell[15].

the model is fast and accurate, assigning a class probability to each grid cell, poses a strong constraint on the number of nearby object detections. A possible drawback of YOLO is, objects that belong to different visual categories and that are centered in a common grid cell might not be correctly classified.

Single Shot Detector (SSD), on the other hand, is a fully convolutional model that makes use of anchor boxes similarly to Faster R-CNN [17]. Anchors are defined over feature maps of consecutive convolutional stages. This allows to efficiently predict bounding boxes at multiple scales. Fu et al. [64] extend the SSD architecture with their so-called deconvolution modules that combine the information from feature maps of various stages via deconvolution and elementwise multiplication to obtain better detection results. Zhang et al. [65] combine SSD with weakly-supervised segmentations to obtain improved results.

CHAPTER 3

VISUAL OBJECT DETECTION WITH LOCAL CONVOLUTIONAL CONTEXT FEATURES

The notion of *context* is frequently encountered in computer vision literature, especially for object detection problem [33, 34, 16, 35, 36, 37, 38]. A context frequently helps recognizing the extent of an object and usually provides a significant clue when determining the class of the object in question. On the other hand, in some cases, such an information does not contain discriminative information for many visual classes. For instance, in the sky, it is more likely to see an object from plane or bird classes, compared to a car object. Any supporting information about the context of the object as sky might help to discriminate between cars and birds, although such a knowledge does not help to discriminate between planes and birds.

CNN architectures that are used in object detection are discriminatively trained on large scale image classification tasks [19, 57, 66, 67]. Such networks that are trained in this manner naturally tend to focus on the most discriminative object features, possibly leaving out the more ambiguous context information. This observation gives the motivation to train a specialized feature extractor for the local context region surrounding the regions of interest as well. Moreover, it could be argued that the role of context becomes even more dominant by challenging samples, such as objects that are captured from certain distance, under weak illumination or strongly occluded.

Context can be local or global depending on whether this information is extracted from neighboring pixels of a region or the whole image containing the object. The local context region can be defined via a rectangular window outside a bounding box. In segDeepM [35], context features are extracted by using a CNN that is finetuned

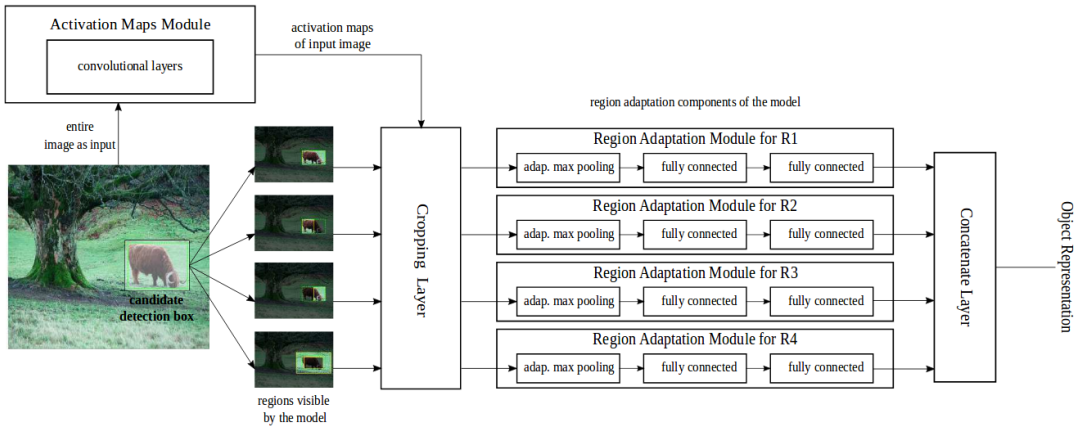

Figure 3.1: Multi-region semantic segmentation-aware CNN model [16]

with expanded versions of ground truth boxes which contain not only the object but also the surrounding context region. In ParseNet [68], global context information is extracted through Global Average Pooling and two types of features are concatenated to obtain smooth and occlusion aware semantic segmentation results. A similar Global Pooling strategy is applied in Deep Residual Networks [67] as well. In the proposed framework, the context features are not global, they are extracted from a local context region defined around each region of interest.

On the other hand, Gidaris and Komodakis [16] train different fully-connected feature stages for different subregions assigned to each region of interest on top of shared convolutional feature maps as depicted on Figure 3.1. Gidaris and Komodakis have a *discriminative feature diversification* approach to enforce different network components learning diversified discriminative features from different subregions assigned to a single proposal region. They report decent detection results even by only using the context features, which suggests that context rings of a region of interests provide strong clues for the task of object detection. In this thesis, a similar approach is followed; however, the proposed method differs from [16] in such a way that the authors learn different fully-connected layers for different subregions on top of a shared convolutional feature map, whereas in this thesis, the convolutional features for different subregions that preserve the spatial relationship after combining are being learned.

In proposal-based detection methods, it is a common practice that once region proposals are obtained, information related to rest of the image is left out. It is argued

that the surrounding features can be further useful for better classification and localization of objects. Context features might render the final bounding box regression stage in Faster R-CNN [17] more purposeful by allowing it to look at a wider range for discovering the full extent of an object. Although the proposed context-based extension is experimented on Faster R-CNN, it might be applicable to other region proposal-based methods.

Faster R-CNN [17] is an end-to-end trainable CNN-based object detection method that performs proposal generation with a fully convolutional network, namely Region Proposal Network (RPN). Proposals are sometimes called Regions of Interest (RoI). In Faster R-CNN, proposal generation, bounding box regression and classification are built upon a shared convolutional feature map. Faster R-CNN is depicted on Figure 3.2.

The method formulates the bounding box regression through anchor boxes, which are reference windows that are hypothesized at various positions, scales and aspect ratios. Ground truth bounding boxes are transformed into regression targets by using the following equations:

$$t_x^* = (x^* - x_a)/w_a \tag{3.1}$$

$$t_y^* = (y^* - y_a)/h_a \tag{3.2}$$

$$t_w^* = log(w^*/w_a) \tag{3.3}$$

$$t_h^* = log(h^*/h_a) \tag{3.4}$$

where x, y, w, h are box center coordinates, width and height; * and $_a$ denote ground truth and anchor, respectively. This transformation is a normalization over the image dimensions and makes the system robust to image scale. The transformation is essential for numerical stability. It should be noted that the inverse of these transformations should be applied to network outputs denoted t (without *) to obtain predicted box coordinates. RPN Regression Loss is defined as

$$L_{RPNreg} = \sum\nolimits_{r \in proposals} p_r^* L_{reg}(r, t_r^*) \tag{3.5}$$

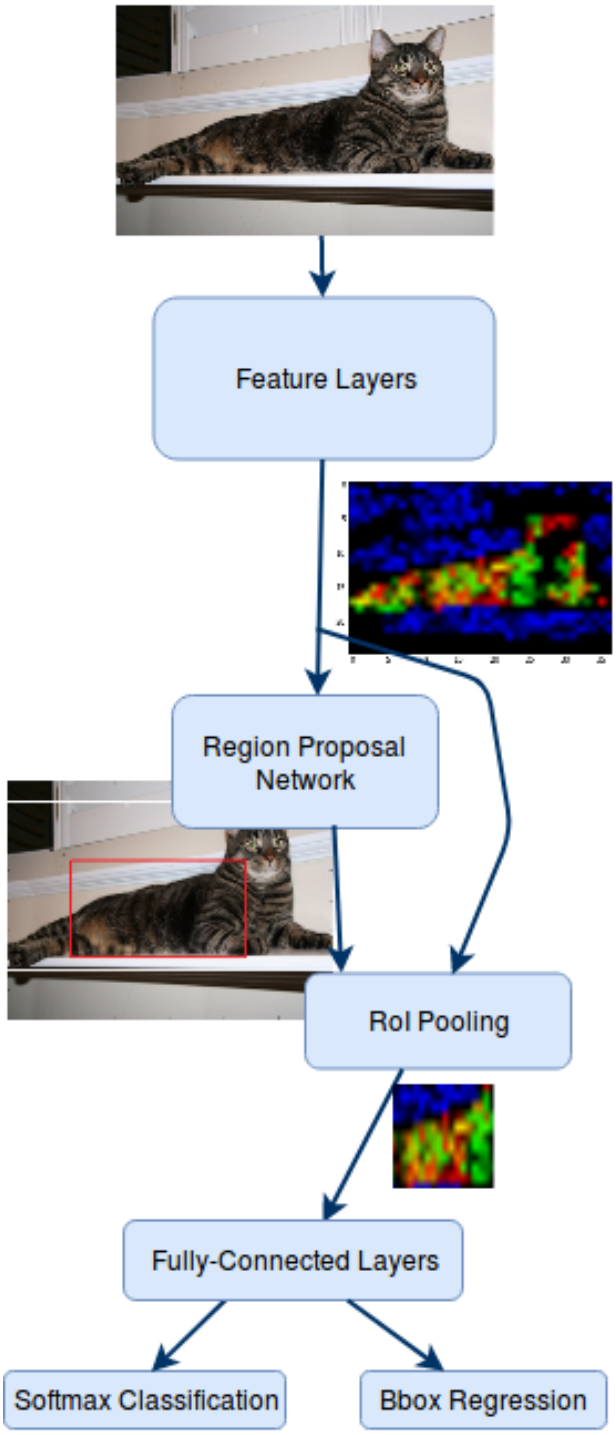

Figure 3.2: Faster R-CNN [17]. Three most activated feature maps are shown in RGB. Red box: Proposal, White box: Ground Truth.

RPN Regression Loss is applied to predictions for positive anchor boxes only. Positive anchors are the anchor boxes that have IoU with some ground truth box higher than a certain threshold. This is expressed in Eqn. 3.5 through the indicator p_r^* that takes the value 1, if the anchor box is positive and 0 otherwise.

In addition to the regression term, there is also a classification term in the RPN loss. For each proposal prediction, RPN returns both the coordinates of the proposal and a corresponding confidence value p_r expressing how confident the system is about this proposal. RPN Classification Loss operates on this confidence, therefore the classification is over two classes: Object vs. No-Object. It should be noted that since RPN is trained with ground truth boxes belonging to a certain set of visual object classes, it is not expected to recognize generic objects. RPN Classification Loss Term is given in Eqn. 3.6.

$$L_{RPNcls} = -\sum\nolimits_{r \in proposals} log p_r \tag{3.6}$$

It should be noted that the confidence values p_r given in Eqn. 3.6 are obtained through softmax over two values produced by the network which is also a crucial step for the system to properly work. Softmax ensures p_r is a value between 0 and 1.

Hence, the resulting overall RPN loss is obtained as

$$L_{RPN} = \lambda_1 L_{RPNreg} + \lambda_2 L_{RPNcls} \tag{3.7}$$

where λ_1 and λ_2 are hyperparameters that determine the trade-off between bounding-box regression and classification.

Final detection loss of Faster R-CNN is defined in a similar manner with two main differences:

1. Classification is over (number of classes + 1) instead of 2. Here, +1 stands for the background class.

2. Ground truth bounding boxes undergo a similar transformation as in Eqn.'s 3.1, 3.2, 3.3, 3.4 but this time with respect to proposals instead of the anchor boxes.

The regression targets for the final detection stage are expressed as follows:

$$t_x^* = (x^* - x_p)/w_p \tag{3.8}$$

$$t_y^* = (y^* - y_p)/h_p \tag{3.9}$$

$$t_w^* = log(w^*/w_p) \tag{3.10}$$

$$t_h^* = log(h^*/h_p) \tag{3.11}$$

Here, subscript p denotes proposals output by RPN. Final detection loss is expressed in Eqn.'s 3.12, 3.13, 3.2.2.

$$L_{DETreg} = \sum\nolimits_{i \in dets} p_i^* L_{reg}(i, t_i^*) \tag{3.12}$$

$$L_{DETcls} = -\sum\nolimits_{i \in dets} log p_{cls,i} \tag{3.13}$$

$$L_{DET} = \lambda_3 L_{DETreg} + \lambda_4 L_{DETcls} \tag{3.14}$$

For both RPN and final detections, bounding box regression is learned through the so-called smooth L1 distance defined in Eqn. 3.16. Smooth L1 distance between regression targets and predictions are summed over four coordinates as in Eqn. 3.15.

$$L_{reg}(b, t_b^*) = \sum\nolimits_{c_b \in (x_b, y_b, w_b, h_b)} smooth_{L1}(t_{c_b} - t_{c_b}^*) \tag{3.15}$$

$$smooth_{L1} = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{3.16}$$

In the original Faster R-CNN [17], an alternating training scheme is applied. In this alternating scheme, firstly RPN is trained applying the RPN loss defined by Eqn. 3.7. In the next step, a Fast R-CNN is trained using the proposals generated by RPN trained in the first step and applying the loss defined in 3.2.2. In the third step, RPN is finetuned on top of the base features that were finetuned in the second step. Finally,

the layers unique to Fast R-CNN are finetuned. Due to practical concerns, we decide to follow the joint training strategy which we find sufficient for our purposes. Joint training reportedly yields close results to the 4-stage alternating training scheme [17]. We apply the following multi-task loss in all our experiments:

$$L_{total} = L_{DET} + L_{RPN} + L_{wd} \tag{3.17}$$

Here, L_{wd} is the weight regularization (decay) term defined as

$$L_{wd} = \sum\nolimits_{w \in network\ parameters} \lambda_w w^2 \tag{3.18}$$

where λ_w is decay coefficient of the variable w.

Total loss expressed in Eqn. 3.17 is minimized by appyling Stochastic Gradient Descent (SGD) with momentum. In SGD, parameter updates are carried out as

$$\theta_{i+1} = \theta_i - \eta \nabla_\theta L_{total}(\theta)|_{\theta=\theta_i} \tag{3.19}$$

where θ is the vector of trainable parameters, θ_i is its value at i'th iteration and η is the learning rate. SGD with momentum extends Eqn. 3.19 with a momentum term to reach convergence faster. Formally, update rule is given as

$$v_i = \gamma v_{i-1} + \eta \nabla_\theta L_{total}(\theta)|_{\theta=\theta_i} \tag{3.20}$$

$$\theta_{i+1} = \theta_i - v_i \tag{3.21}$$

where γ is the momentum, a value between 0 and 1.

3.1 Proposed Method with Local Convolutional Context Features

In the proposed method, local context of a region proposal is utilized to improve detection results. Local context region is defined through a context box that is obtained

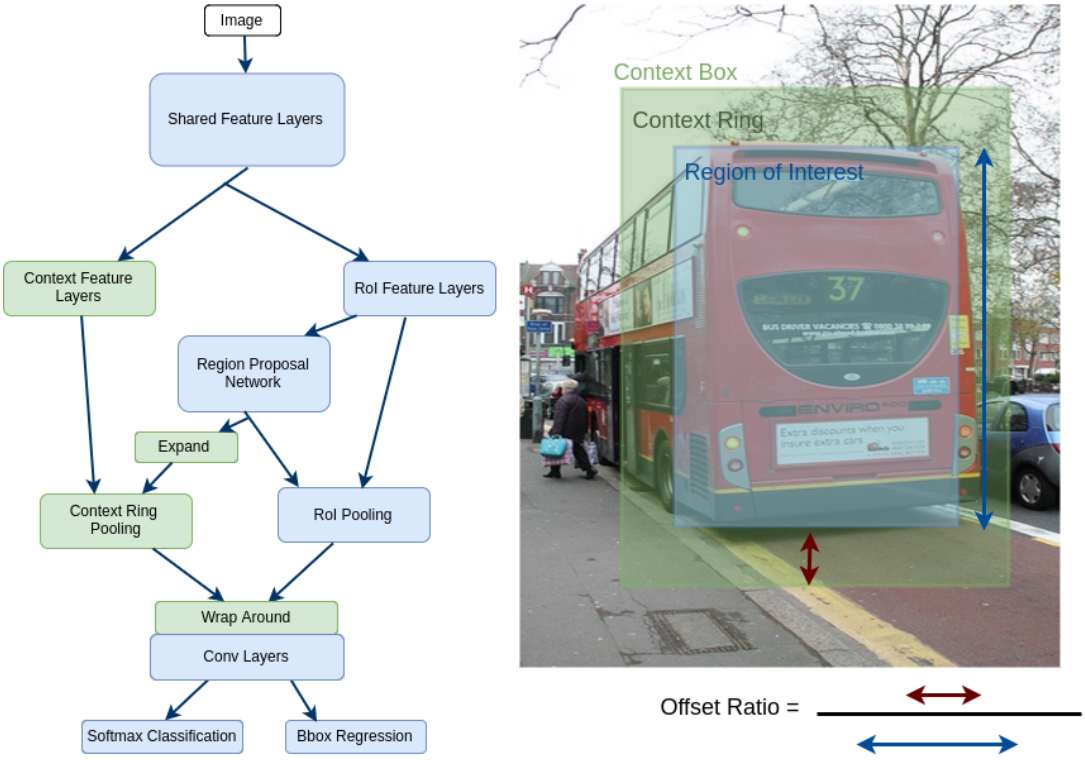

Figure 3.3: Overview of the Proposed Algorithm. Layers shown in green are proposed for improving detection performance. (Best viewed in color)

by expanding the RoI box by a certain ratio as depicted on Figure 3.3. This ratio, denoted as *offset ratio*, determines the extent of surrounding features to be utilized. The region between a bounding box and the corresponding context box is called the context ring. To illustrate, if the object region has a shape of 100x100 pixels and the corresponding context region is chosen to be 120x120 pixels the offset ratio is 0.1. On top of a shared feature extraction stage, there are RoI and context feature layers that aim to learn diversified features for the context region and the region of interest. Both feature extractors are identical in their structures and they are initialized by the same weights learned from Imagenet [40]. Figure 3.3 presents an overview of the proposed algorithm.

In the proposed model, there are two adaptive pooling layers: One of them is the RoI pooling layer that pools features from a region of interest in the exact same way Faster R-CNN [17] does. The newly introduced one is called *context ring pooling layer*. Context ring pooling layer pools context features to generate a fixed size representation for the local context. Context ring pooling is a two-stage operation that

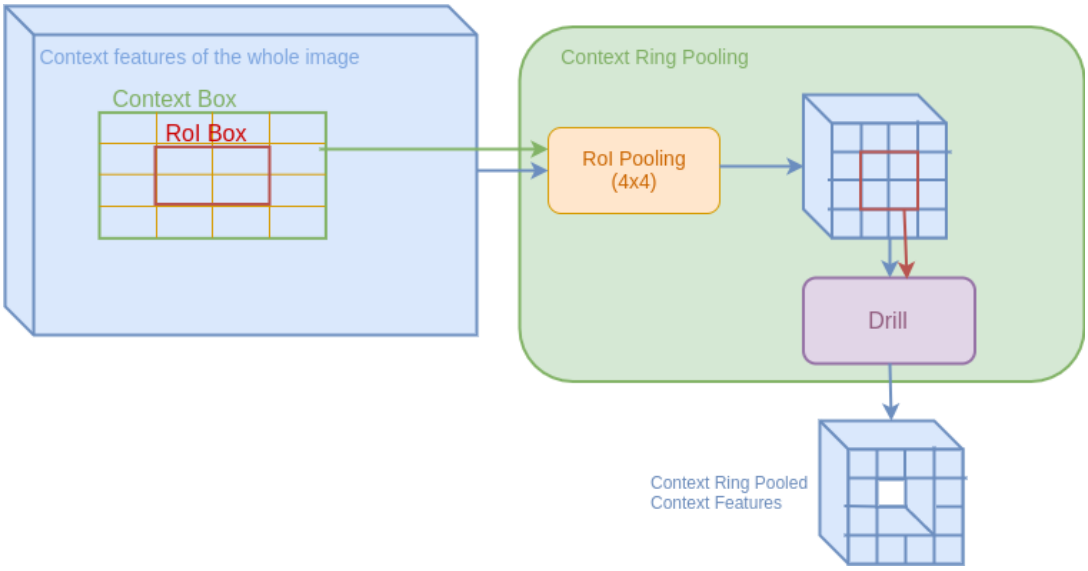

Figure 3.4: Context Ring Pooling. In this illustration, context poolsize is 4x4 and roipool size is 2x2. Offset Ratio is 0.5.

consists of RoI pooling over the context box followed by a so-called *Drill* operation that drills out the inner features and generates a ring-shaped fixed size representation for the context region. Roipool size of the RoI pooling operation which is performed inside the context ring pooling layer is called the *contextpool size*. Context ring pooling is illustrated in Figure 3.4. The relationship between offset ratio, roipool size and contextpool size is expressed as follows:

$$Offset\ Ratio = \frac{(contextpool\ size) - (roipool\ size)}{2 * (roipool\ size)} \tag{3.22}$$

A novel *wrap around* operation lies at the heart of the proposed method (The operation of wrap around layer is depicted in Figure 3.5). Wrap around combines the RoI pooled features with the context ring pooled features in a special way that the spatial relationships between the features are preserved. The output of the wrap around layer is fed to the subsequent stages of the network. By relating the three parameters as in Eqn. 3.22, we ensure that the context ring pooled and RoI pooled features have the same scale which might be crucial since they are concatenated together and fed into convolutional filters. It should be noted that after the proposed modifications, the resulting system remains to be end-to-end trainable.

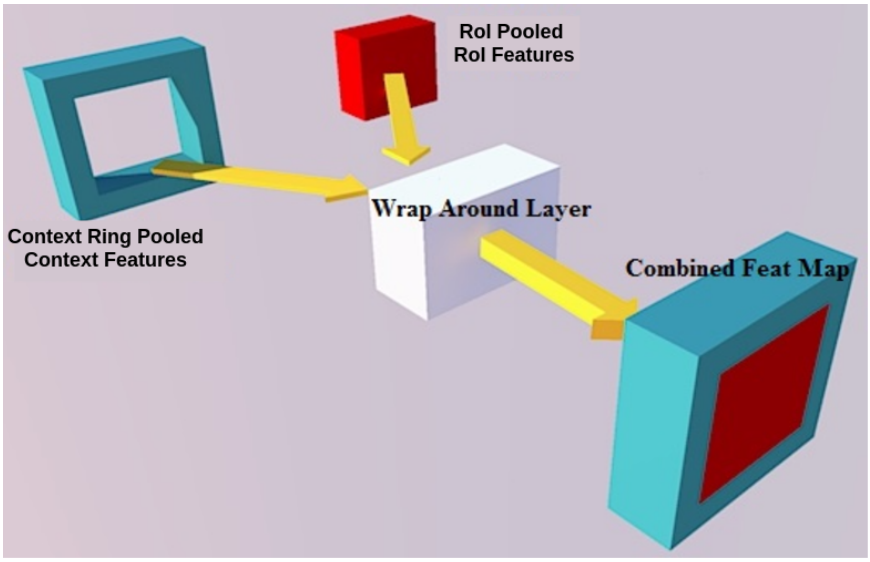

Figure 3.5: Wrap Around Layer combines two feature maps while preserving the spatial relationship between them.

3.2 Experimental Work

In this section, we present the experimental work. Section 5.3.1 describes the dataset and the evaluation procedure. Section 3.2.2 presents detailed information related to training scheme. Section 5.3.3 presents the conducted experiments and their results.

3.2.1 Datasets and Evaluation Metrics

The experiments are conducted with PASCAL VOC 2007 [18], which is a popular object detection benchmark dataset. Pascal VOC 2007 contains a total of 9963 photographs containing at least 1 instance of 20 objects. PASCAL VOC object categories are as follows: Aeroplane, Bicycle, Bird, Boat, Bottle, Bus, Car, Cat, Chair, Cow, Dining Table, Dog, Horse, Motorbike, Person, Pottedplant, Sheep, Sofa, Train and TV/Monitor.

PASCAL VOC is split into training+evaluation (train+val) and test sets containing 5011 and 4952 images, respectively. All the presented experiments use train+val set as the training set and the test results are obtained with the test set. The number of instances and images per category in VOC 2007 is plotted on Fig. 3.6.

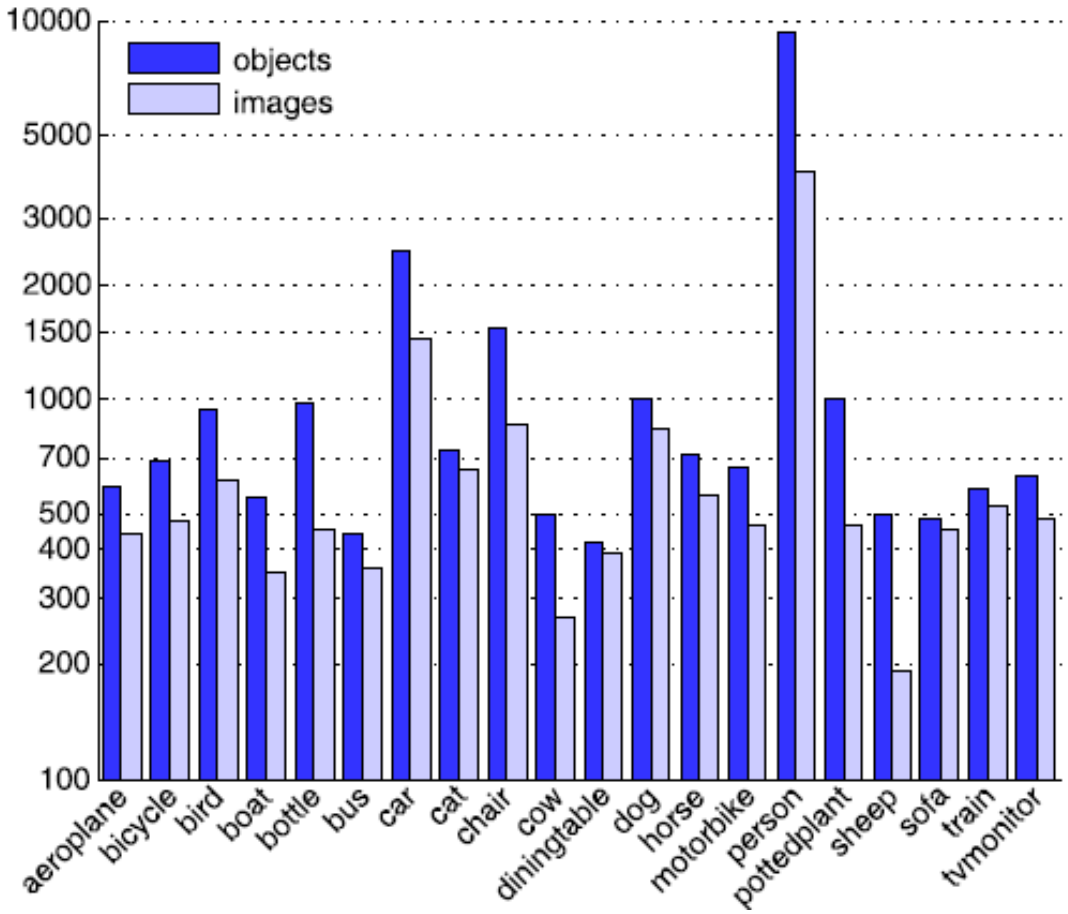

Figure 3.6: Number of instances and images per category in PASCAL VOC 2007 (taken from [18]]

For assessing the performance of detection algorithms, *precision* and *recall* are two important metrics. In order to have a grasp of these two concepts, one can explain by a toy example: Assume that a sheep detector algorithm is applied to a photograph of sheep. Let there be 5 sheep in the photograph. The tested sheep detector is capable of detecting 1000 sheep and returns confidences associated with every 1000 detections. 1000 is a quite large number and most of the time, it is reasonable to apply a confidence threshold to allow for the most confident detections only. Let us assume 6 detections surpass the threshold and among these detections, only 4 detections have sufficient overlap with individual sheep instances in the photograph. These 4 detections are called *true positives*. Remaining two detections are wrong and they are called *false positives*. In the photograph there were 5 sheep and the algorithm could detect only four of them. The sheep that is missed is called a *false negative*.

More formally, precision is defined as

$$Precision = \frac{TP}{TP + FP} \tag{3.23}$$

while recall is given by the following relation:

$$Recall = \frac{TP}{TP + FN} \tag{3.24}$$

where TP is the number of true positives, FP is the number of false positives and FN is the number of false negatives. For a detection algorithm, it is desirable that both precision and recall are high. At test time, one adjusts a parameter called confidence threshold. Precision and recall are significantly affected by this parameter. Precision is likely to be high and recall is low when the threshold is set quite high and vice versa. The system moves on a precision-recall curve as depicted on Figure 3.7 when the confidence threshold is varied. Confidence threshold is merely a means to select an operating point on this curve. The whole curve is much more informative about the performance of the algorithm when compared to a single point on the curve.

Detection performance can also be measured with *Average Precision* (AP). AP is a summary of the precision-recall curve of a single class. In official PASCAL VOC

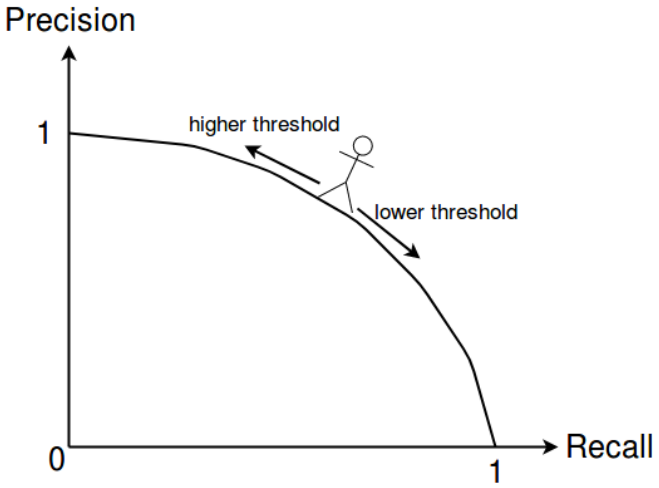

Figure 3.7: Precision-Recall Curve.

evaluation, this summary is taken by interpolating the precision-recall curve at 11 recall levels [18] as follows:

$$AP_c = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \max_{r^*: r^* \geq r} p_c(r^*) \tag{3.25}$$

where p_c is the precision-recall curve for class c. Mean of the Average Precisions over all classes is called *mean Average Precision* (mAP):

$$mAP = \frac{1}{\#ofclasses} \sum_{c \in classes} AP_c \tag{3.26}$$

3.2.2 Training

All training is performed using the machine learning library Tensorflow [69] on a NVIDIA GTX 980 graphics card. Shared convolutional feature layers are kept fixed (no backpropagation). Loss component weights $\lambda_1 - \lambda_4$ in Eqn.'s 3.7, are all set to 1. Stochastic Gradient Descent (SGD) with momentum equal to 0.9 is applied throughout the whole experimentation. During training, the images are fed in single scale with the shorter side is 400 (In the original paper [17], this value was equal to 600) and the aspect ratios are kept the same. For pretrained layers, weight decay coefficient λ_w is set to 0.0005, for newly initialized layers λ_w is set to 0.00025. During

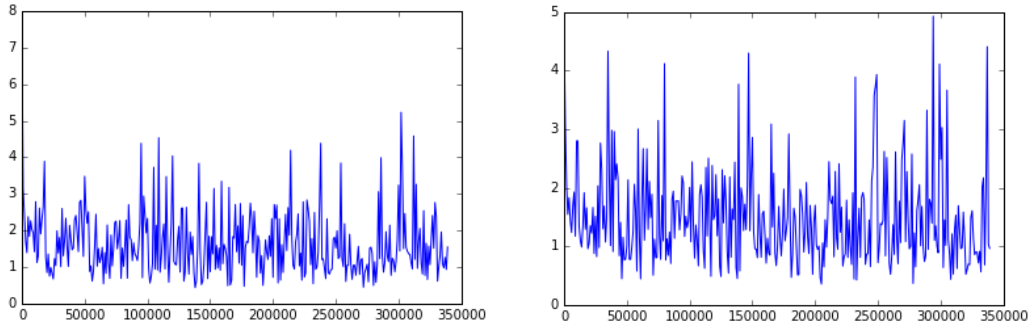

Figure 3.8: Test Loss vs. Iteration for Models B_5 (left) and C_{20} (right).

training both the original images and their left-right flipped versions are fed; no other data augmentation is applied. Initially learning rate is 0.0001 and decays by 0.8 after every 25k iterations. Since the models are trained with single image batches, loss curves are typically quite noisy and it is impractical to apply early stopping based on training and test losses. Test loss curves for a baseline model and a context model are given on Figure 3.8. For a fair comparison, we fixed the number of training iterations to 340k for all models.

Training is performed on single image multiple RoI batches. Following [17], we applied a batch size of 256 consisting of up to 128 positive and down to 128 negative RoIs belonging to a single image during all experiments. During training and test, 10k proposals with the highest confidence generated by RPN are applied Non-Max Suppression (NMS) and up to 2k of the remaining proposals are used.

3.2.3 Experiments

In this section, several experiments are performed to investigate the effects of design considerations when constructing a context model. There are two important design selections regarding the context extension. These are the number of context and RoI feature layers and the offset ratio. Experiments 1 and 2 deal with context feature layers and Experiment 3 investigates the effect of the offset ratio.

In all experiments, VGG-16 [19] architecture pretrained on Imagenet [40] is employed for feature extraction. The feature extractor network is depicted on Fig. 3.9.

Table 3.1: Number of parameters of baseline and context models

Model Name	**Number of Parameters (in Millions)**
Original Faster R-CNN with VGG-16	137.08
B_1	22.04
C_1	25.04
C_{f2}	27.40
C_{f3}	29.76
C_{33}	26.12
B_7	22.30
C_{14}	25.53
C_{29}	26.82
B_5	21.88
C_{20}	24.67
B_8	22.68
C_{12}	26.12
C_{25}	27.62
B_9	23.17
C_{22}	28.54

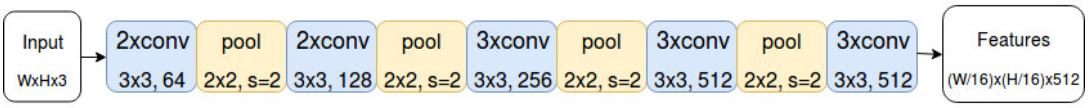

Figure 3.9: Feature extraction architecture used: VGG-16 [19]

In baseline models, which are denoted with B and BC, this architecture is used without any modification. In context models, which are denoted with C, the final layers are branched into context and RoI feature layers as depicted on Figures 3.12,3.13 and 3.14.

Original Faster R-CNN architecture has large fully connected stages after the RoI pooling stage. Due to limited hardware resources, a simplified version is utilized by replacing the two fully-connected 4096 neuron layers in the original model with two 3x3x512 convolutional layers. With this modification, around 115 million trainable parameters are saved. Number of parameters for all trained models are presented on Table 3.1.

Table 3.2: Models trained in Experiment 1

Model	roipool(/ctxpool)	# of Ctx Layers	Pooled Regions	Offset Ratio
B_1	6x6	0	RoI	-
BC_1	8x8	0	RoI and Ctx Ring	0.17
C_1	6x6/8x8	1	RoI and Ctx Ring	0.17
B_7	7x7	0	RoI	-
BC_{14}	9x9	0	RoI and Ctx Ring	0.14
C_{14}	7x7/9x9	1	RoI and Ctx Ring	0.14
B_5	5x5	0	RoI	-
BC_{20}	7x7	0	RoI and Ctx Ring	0.20
C_{20}	5x5/7x7	1	RoI and Ctx Ring	0.20

3.2.3.1 Experiment 1: Separate Feature Extractors

In this experiment, we compare two different scenarios for exploiting local context information. Models C_1, C_{14} and C_{20} are context models that have separate feature extraction stages for the region of interest and the context ring. Models BC_1, BC_{14} and BC_{20} pool features from the context box but they do not have separate feature extraction stages for the context ring and the region of interest. Models B_1, B_7 and B_5 are baseline models that pool features from the region of interest only. Our aim is to investigate whether it is essential to learn diversified features for the context region and the region of interest by comparing models denoted with B, BC and C. The experiment is repeated for three different offset ratios (0.14, 0.17, 0.20) to reduce the possibility of having coincidental results. Model roipool sizes and offset ratios are presented in Table 3.2. Model architectures are presented in Figure 3.11.

Experimental results for different visual categories are presented on Table 3.3. Comparing results for models denoted with B, BC and C; it can be concluded that although by simply expanding the region of interest into a local context region, as in models BC, some improvements over baseline models (denoted with B) are observed, separate feature extractors for the context region and the region of interest lead to best improvements in detection performance. Visual results for baseline model B1 and context model C1 are given on Figure 3.10.

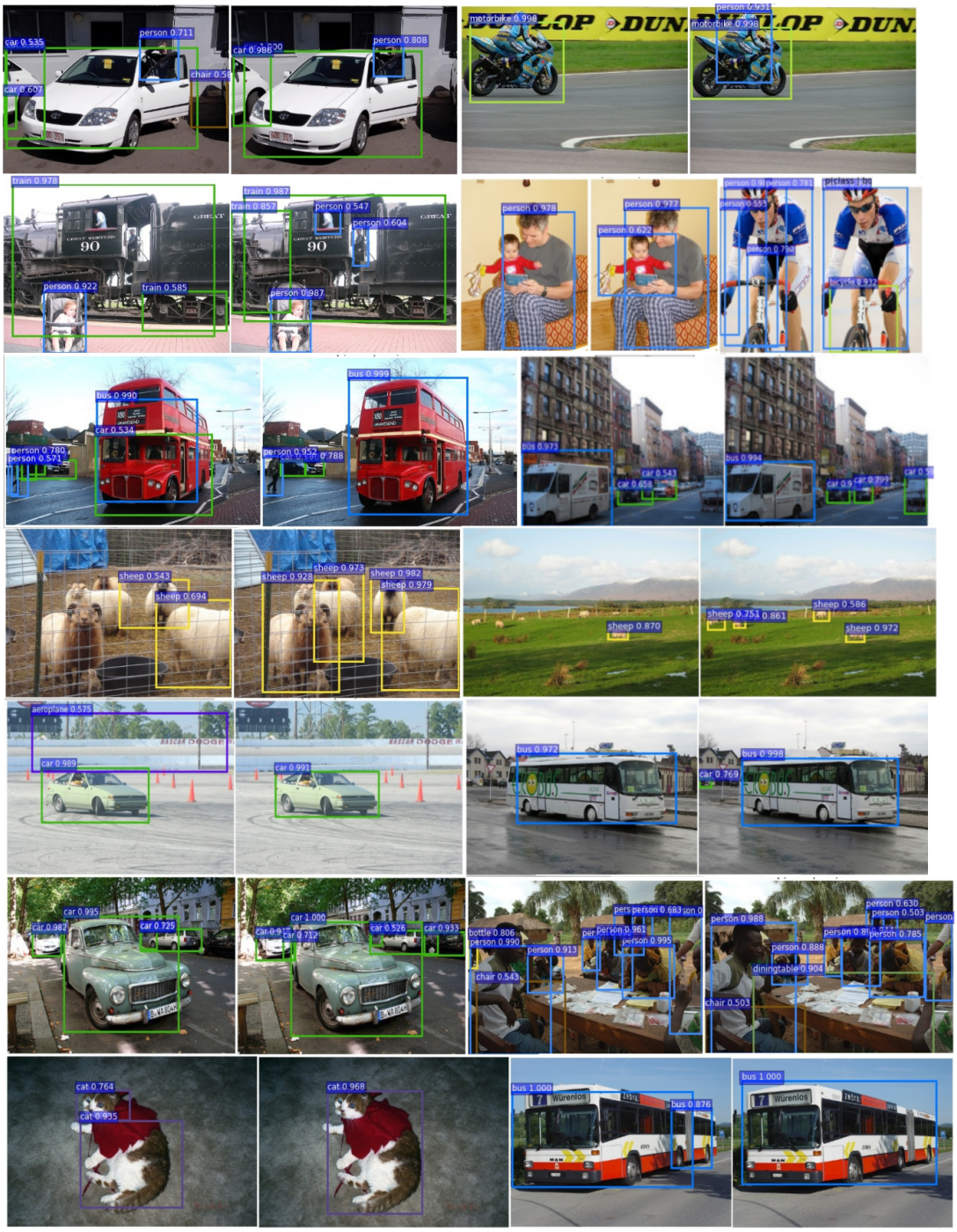

Figure 3.10: Visual results for baseline model B_1 (left) and context model C_1 (right). Confidence Threshold = 0.5.

Table 3.3: AP results of Experiment 1 over VOC 2007 Test Set

Class	B_1	BC_1	C_1	B_7	BC_{14}	C_{14}	B_5	BC_{20}	C_{20}
a.plane	58.94	61.29	**67.67**	**60.58**	59.84	59.79	58.56	62.29	**63.36**
bicycle	69.78	73.34	**76.42**	71.83	**73.88**	73.69	72.12	72.94	**73.35**
bird	55.60	55.46	**61.13**	56.45	54.88	**59.70**	55.22	54.09	**58.68**
boat	38.54	45.32	**54.85**	39.07	38.87	**41.93**	39.46	40.47	**41.37**
bottle	30.10	33.10	**34.09**	27.03	29.41	**29.56**	28.25	31.23	**32.27**
bus	67.96	69.78	**73.81**	**68.08**	65.73	67.71	64.87	**71.92**	69.68
car	68.28	67.96	**74.68**	67.45	66.95	**70.93**	67.22	67.76	**70.80**
cat	74.46	75.06	**80.32**	73.28	73.31	**77.85**	73.58	75.07	**76.06**
chair	38.15	37.44	**40.30**	36.79	**37.26**	37.04	35.38	35.47	**38.36**
cow	61.37	62.26	**71.13**	62.04	63.95	**64.46**	58.00	64.10	**64.45**
d.table	55.46	58.01	**59.75**	56.07	**58.61**	56.75	52.49	**61.46**	58.22
dog	72.91	74.08	**76.35**	71.24	71.17	**74.11**	72.90	72.09	**75.04**
horse	75.23	74.91	**79.36**	76.32	75.71	**76.68**	74.66	76.63	**77.27**
m.bike	70.00	71.48	**75.84**	71.93	69.79	**73.11**	66.26	68.68	**71.67**
person	64.93	64.33	**68.11**	64.38	63.70	**66.26**	63.57	64.36	**66.31**
p.plant	27.59	27.15	**31.13**	**31.00**	29.50	29.88	26.03	26.29	**28.81**
sheep	54.24	53.37	**63.26**	52.89	**54.52**	51.84	48.93	**55.58**	54.61
sofa	60.08	57.88	**62.65**	59.64	56.41	**60.95**	56.31	58.37	**63.71**
train	72.35	70.47	**73.63**	70.86	68.60	**71.61**	70.97	71.05	**73.51**
tv	**62.80**	60.88	62.48	59.38	60.67	**61.75**	61.28	60.63	**62.24**
mean	58.94	59.68	**64.35**	58.82	58.64	**60.28**	57.30	59.52	**60.99**

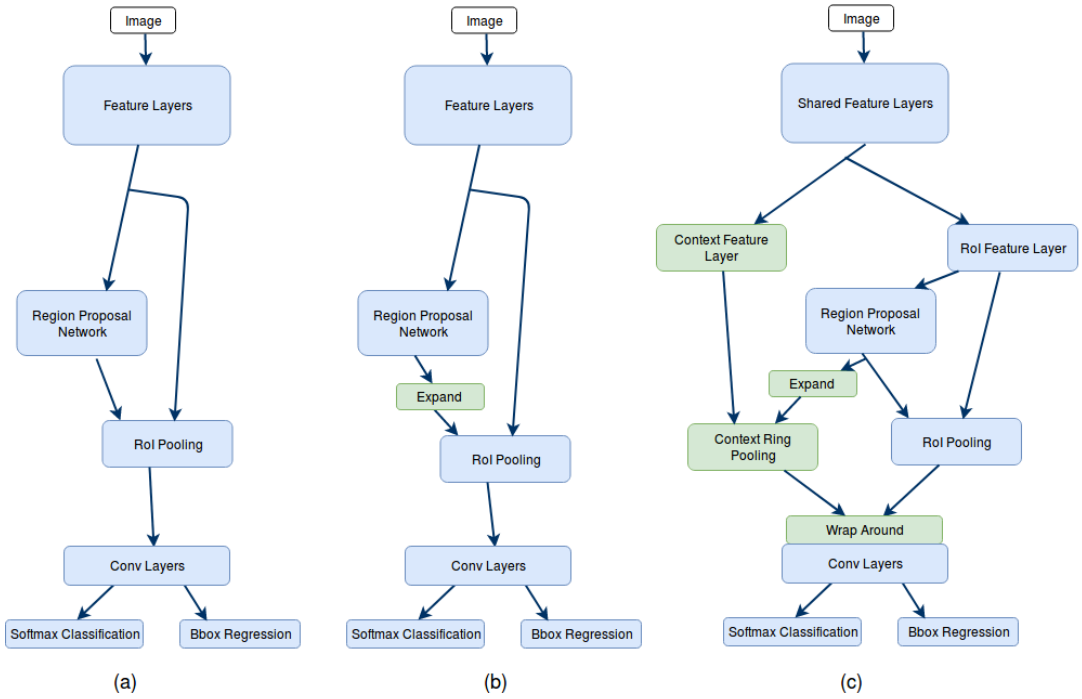

Figure 3.11: Model architectures in Experiment 1: (a) B_1, B_7, B_5 (b) BC_1, BC_{14}, BC_{20} (c) C_1, C_{14}, C_{20}

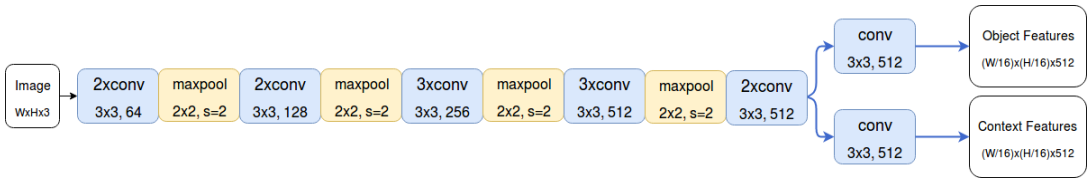

Figure 3.12: Feature extractor of Model C_1

3.2.3.2 Experiment 2: Number of Separate Feature Extraction Stages

In order to observe the effect of having different number of context feature layers, the models C_1, C_{f2} and C_{f3} are trained by 1, 2 and 3 context feature layers, respectively. Feature extraction stages of these models are depicted on Figures 3.12, 3.13 and 3.14. Properties of the models are summarized in Table 3.4

Experimental results presented in Table 3.5 suggest that the number of separate context and RoI feature layers has a significant effect on the performance of the algorithm. Best results are obtained with a single context feature layer (C_1). There are improvements with respect to the baseline with 2 (C_{f2}) and 3 (C_{f3}) layers, although less when compared to C_1. It can be argued that as the number of unshared layers in-

Table 3.4: Properties of baseline and context models in Experiment 2

Model	**roipool(/ctxpool)**	**# of Ctx Layers**	**Pooled Regions**	**Offset Ratio**
B_1	6x6	0	RoI	-
C_1	6x6/8x8	1	RoI and Ctx Ring	0.17
C_{f2}	6x6/8x8	2	RoI and Ctx Ring	0.17
C_{f3}	6x6/8x8	3	RoI and Ctx Ring	0.17

Table 3.5: APs over VOC 2007 Test Set for Experiment 2

Class	B_1	C_1	C_{f2}	C_{f3}
aeroplane	58.94	**67.67**	60.58	61.20
bicycle	69.78	**76.42**	70.28	69.61
bird	55.60	**61.13**	56.18	57.00
boat	38.54	**54.85**	40.25	42.44
bottle	30.10	34.09	**35.36**	30.62
bus	67.96	**73.81**	67.45	70.36
car	68.28	**74.68**	69.19	70.77
cat	74.46	**80.32**	74.41	77.47
chair	38.15	**40.30**	34.68	36.03
cow	61.37	**71.13**	62.07	64.42
diningtable	55.46	58.01	**59.75**	56.73
dog	72.91	**76.35**	71.33	72.23
horse	75.23	**79.36**	76.65	75.89
motorbike	70.00	**75.84**	68.87	70.83
person	64.93	**68.11**	64.88	65.52
pottedplant	27.59	**31.13**	26.41	29.38
sheep	54.24	**63.26**	51.91	53.40
sofa	60.08	62.65	59.72	**63.33**
train	72.35	**73.63**	73.61	73.44
tv/monitor	62.80	62.48	**64.15**	62.42
mean	58.94	**64.35**	59.24	60.18

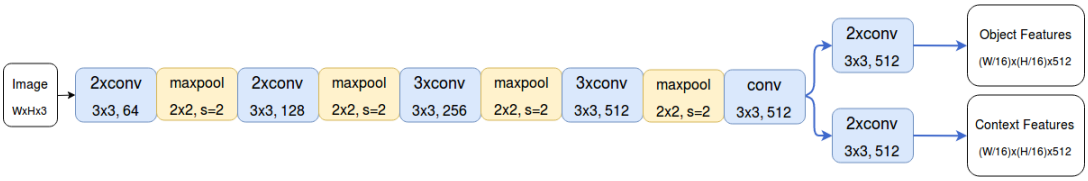

Figure 3.13: Feature extractor of Model C_{f2}

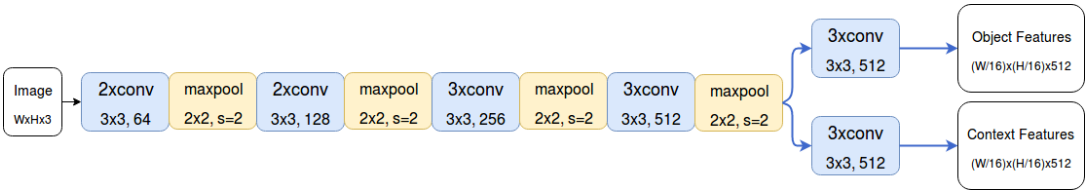

Figure 3.14: Feature extractor of Model C_{f3}

crease, the complexity of the problem also increases and it becomes harder to obtain improved results with the same number of iterations.

3.2.3.3 Experiment 3: Offset Ratio

In this experiment, we train models with different offset ratios in order to observe the effect of offset ratio. Offset ratio, contextpool size and roipool size are related to each other as expressed in Eqn. 3.22. To illustrate, if contextpool and roipool sizes are chosen as 8x8 and 6x6, then the offset ratio becomes 0.17. Since contextpool size and roipool size have to be integers, one cannot arbitrarily adjust the offset ratio. In order to perform fine adjustment on the offset ratio, both roipool size and contextpool size need to be changed. However, it should be noted that roipool size is a parameter for the baseline model as well. Since roipool size might be decisive for the perfomance of the baseline model, one should also train the respective baseline models having the same roipool size with each context model to have a fair comparison. Based on this reasoning, the context models C_1 and C_{33} are to be compared against the baseline model B_1; context models C_{14} and C_{29} are to be compared with the corresponding baseline model B_7; context model C_{20} is to be compared with the corresponding baseline model B_5; context model C_{12} and C_{25} are to be compared with the corresponding baseline model B_8. Model properties are summarized in Table 3.6. During context pooling, feature maps are padded with the mean value of the

Table 3.6: Properties of baseline and context models in Experiment 3

Model	roipool(/ctxpool)	# of Ctx Layers	Pooled Regions	Offset Ratio
B_1	6x6	0	RoI	-
C_1	6x6/8x8	1	RoI and Ctx Ring	0.17
C_{33}	6x6/10x10	1	RoI and Ctx Ring	0.33
B_7	7x7	0	RoI	-
C_{14}	7x7/9x9	1	RoI and Ctx Ring	0.14
C_{29}	7x7/11x11	1	RoI and Ctx Ring	0.29
B_5	5x5	0	RoI	-
C_{20}	5x5/7x7	1	RoI and Ctx Ring	0.20
B_8	8x8	0	RoI	-
C_{12}	8x8/10x10	1	RoI and Ctx Ring	0.125
C_{25}	8x8/12x12	1	RoI and Ctx Ring	0.250
B_9	9x9	0	RoI	-
C_{22}	9x9/13x13	1	RoI and Ctx Ring	0.22

feature map where necessary. In baseline models, green layers in Fig. 3.3 are left out so that only the blue layers are active. Experimental results for Experiment 3 are presented in Tables 3.7-3.8.

For most of the experiments, context models outperform the corresponding baselines. Among context models that have different offset ratios, there are significant differences in performance as evident from Tables 3.7-3.8. Performance difference between different context models suggests that offset ratio critically effects the performance of a context-based model. For models C_{12} and C_{14}, which have the smallest offset ratios, the improvements are less significant when compared to models C_1, C_{20}, C_{22}, C_{25}, C_{29}. The worst result is obtained with the highest offset ratio model (C_{33}). This result might be due to the fact that spatially distant features provide irrelevant information. It should also be noted that since the image dimensions are bounded, a high offset ratio results in more padding in pooled context features.

3.3 Comparison with Similar Methods

Results of our best performing context model (C_1) are compared with two proposal-based detection methods (Faster R-CNN [17] and Multi-region segmentation-aware

Table 3.7: APs over VOC 2007 Test Set for Experiment 3

Class	B_1	C_1	C_{33}	B_7	C_{14}	C_{29}
aeroplane	58.94	**67.67**	57.95	60.58	59.79	**66.03**
bicycle	69.78	**76.42**	69.59	71.83	73.69	**73.84**
bird	55.60	**61.13**	53.71	56.45	59.70	**61.16**
boat	38.54	**54.85**	39.73	39.07	41.93	**48.34**
bottle	30.10	**34.09**	29.19	27.03	29.56	**35.11**
bus	67.96	**73.81**	69.97	68.08	67.71	**74.03**
car	68.28	**74.68**	68.07	67.45	70.93	**74.55**
cat	74.46	**80.32**	73.50	73.28	77.85	**80.62**
chair	38.15	**40.30**	34.02	36.79	37.04	**39.79**
cow	61.37	**71.13**	60.18	62.04	64.46	**67.45**
diningtable	55.46	**59.75**	50.29	56.07	56.75	**63.40**
dog	72.91	**76.35**	70.40	71.24	74.11	**76.40**
horse	75.23	**79.36**	72.70	76.32	76.68	**78.16**
motorbike	70.00	**75.84**	68.89	71.93	73.11	**74.78**
person	64.93	**68.11**	61.43	64.38	66.26	**67.68**
pottedplant	27.59	**31.13**	27.89	**31.00**	29.88	29.59
sheep	54.24	**63.26**	52.55	52.89	51.84	**61.49**
sofa	60.08	62.65	**64.50**	59.64	60.95	**65.00**
train	72.35	**73.63**	67.63	70.86	71.61	**75.51**
tv/monitor	**62.80**	62.48	61.22	59.38	61.75	**64.86**
mean	58.94	**64.35**	57.67	58.82	60.28	**63.89**
Offset Ratio	-	0.17	0.33	-	0.14	0.29

Table 3.8: APs over VOC 2007 Test Set for Experiment 3

Class	B_5	C_{20}	B_8	C_{12}	C_{25}	B_9	C_{22}
aeroplane	58.56	**63.36**	58.47	60.69	**65.12**	59.61	**62.76**
bicycle	72.12	**73.35**	72.36	71.43	**72.55**	71.22	**72.90**
bird	55.22	**58.68**	56.23	**58.36**	55.80	52.75	**54.17**
boat	39.46	**41.37**	41.20	41.32	**42.06**	35.21	**44.86**
bottle	28.25	**32.27**	29.05	32.90	**33.13**	31.85	**34.34**
bus	64.87	**69.68**	64.29	66.69	**69.52**	66.34	**71.58**
car	67.22	**70.80**	68.80	69.92	**71.04**	68.80	**70.12**
cat	73.58	**76.06**	74.83	75.72	**75.94**	74.07	**77.28**
chair	35.38	**38.36**	36.97	36.70	**36.99**	34.18	**37.25**
cow	58.00	**64.45**	**62.46**	61.73	60.85	59.02	**63.12**
diningtable	52.49	**58.22**	55.34	57.23	**58.36**	53.44	**59.42**
dog	72.90	**75.04**	69.71	72.82	**73.28**	71.04	**73.43**
horse	74.66	**77.27**	**76.04**	75.14	75.23	76.16	**77.01**
motorbike	66.26	**71.67**	69.75	**72.77**	71.30	**71.11**	69.79
person	63.57	**66.31**	65.48	**65.96**	65.89	65.41	**65.55**
pottedplant	26.03	**28.81**	27.11	29.07	**30.09**	27.35	**28.31**
sheep	48.93	**54.61**	53.86	**53.93**	52.29	52.12	**55.53**
sofa	56.31	**63.71**	56.63	63.17	**65.22**	59.83	**60.17**
train	70.97	**73.51**	70.38	72.63	**73.86**	70.77	**71.26**
tv/monitor	61.28	**62.24**	60.82	62.79	**63.88**	62.78	**63.42**
mean	57.30	**60.99**	58.49	60.05	**60.62**	58.15	**60.61**
Offset Ratio	-	0.20	-	0.125	0.25	-	0.22

Table 3.9: Comparison with methods in the literature over VOC 2007 Test Set

Class	**FRCNN [17]**	**MRCNN [16]**	C_1**(ours)**
aeroplane	70.0	**74.9**	67.7
bicycle	**80.6**	75.7	76.4
bird	**70.1**	64.5	61.1
boat	**57.3**	54.9	54.9
bottle	**49.9**	44.7	34.1
bus	**78.2**	74.1	73.8
car	**80.4**	75.5	74.7
cat	**82.0**	76.0	80.3
chair	**52.2**	48.1	40.3
cow	**75.3**	72.4	71.1
diningtable	67.2	**67.4**	58.0
dog	**80.3**	76.5	76.4
horse	**79.8**	72.4	79.4
motorbike	75.0	74.9	**75.8**
person	**76.3**	61.7	68.1
pottedplant	**39.1**	34.8	31.1
sheep	**68.3**	61.7	63.3
sofa	**67.3**	64.0	62.7
train	**81.1**	73.5	73.6
tv/monitor	67.6	**76.0**	62.5
mean	**69.9**	66.2	64.4
# net params (in Millions)	137.08	1210.26	25.04

CNN (MRCNN) [16]) in Table 3.9. All three models are trained on PASCAL VOC 2007 train+val set and they are built on top of VGG-16 feature extraction backbone. MRCNN uses Selective Search [3] for region proposal generation. It should be noted that, although context model C_1 does not outperform the other two methods (except one visual category), it yields comparable results with a much simpler training strategy, smaller input images during both training and test and significantly less number of trainable parameters (given in the last row of Table 3.9).

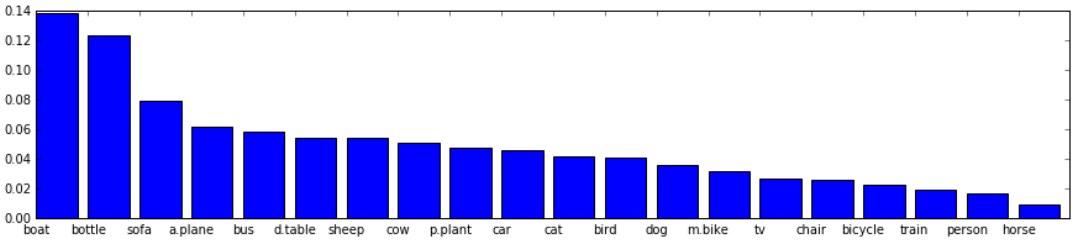

Figure 3.15: Mean Percent Increase in Average Precision vs. visual category.

3.4 Category-level Improvements Achieved by Local Convolutional Context Features

In order to observe the category-level effects of local context features, we define Percent Increase in Average Precision (PIAP) of a context model C as

$$PIAP(C) = \frac{AP(C) - BaselineAP(C)}{BaselineAP(C)}, \tag{3.27}$$

where AP(C) is the Average Precision of model C and Baseline AP(C) is the Average Precision of the corresponding baseline model. PIAP aims to quantify the improvement achieved with context feature extension. By evaluating mean of PIAPs over all trained context models for each visual category, we compare the improvements for different visual categories in Figure 3.15. According to Figure 3.15, best improvements are obtained for visual categories with distinctive context such as boat, bottle, sofa and aeroplane. It should be noted that for certain visual categories such as aeroplane-bird, sheep-cow, cat-dog pairs and land vehicles (car-bus-bicycle-motorbike) object contexts are similar which might confuse the classifier. On the other hand, for the boat category, such similar context categories do not exist therefore it is reasonable to observe a high improvement. In PASCAL VOC, person category does not have a distinctive context therefore it is reasonable to observe a small improvement. Nevertheless, it should also be emphasized that our method exploits the context of *region proposals* instead of objects. The overlaps between context regions of proposals and objects are high as far as region proposals are accurate. The above discussion assumes that there is some correspondence between the contexts of proposals and ground truth objects.

3.5 Feature Diversification and Sparsity

In order to reach an understanding about the operation of the newly proposed system, we visualize object and context features during training. On Figure 3.16, three most activated feature maps (that have the highest average value) from object and context feature layers are shown in Red, Green and Blue channels. It should be noted that "the most activated" does not necessarily mean "the most important" or "the most discriminative". In fact, most activated features tend to be the least sparse features. From Figure 3.16, it is evident that the network is able to learn diversified features for the region of interest and the context region. Also, it is observed that as the training continues, context features become less and less sparse when compared to object feature maps.

3.6 Diagnostic Analysis of Errors

Average precision (AP) provides a convenient way to assess the performance of object detectors. On the other hand, knowing AP alone does not provide much of an insight on what the algorithm is capable of and what it is not. With this motivation, Hoiem et al. [70] developed certain tools for an in-depth analysis of object detectors. In this section, we apply these tools to a baseline and a context model and provide a detailed comparative analysis of their performances.

3.6.1 Analysis of False Positives

For the case of false positives, Hoiem et al. [70] define three types of errors: "localization", "confusion with similar object", "confusion with dissimilar object" and "confusion with background".

Localization error is the case of having a detection with true class but not having sufficient overlap with the ground truth box. IoU of ground truth and detection boxes is between 0.1 and 0.5. PASCAL VOC classes are grouped into super-classes as vehicles, animals and furniture. Classes belonging to the same super-class are considered similar classes. In addition to the super-classes mentioned above, aeroplane and bird

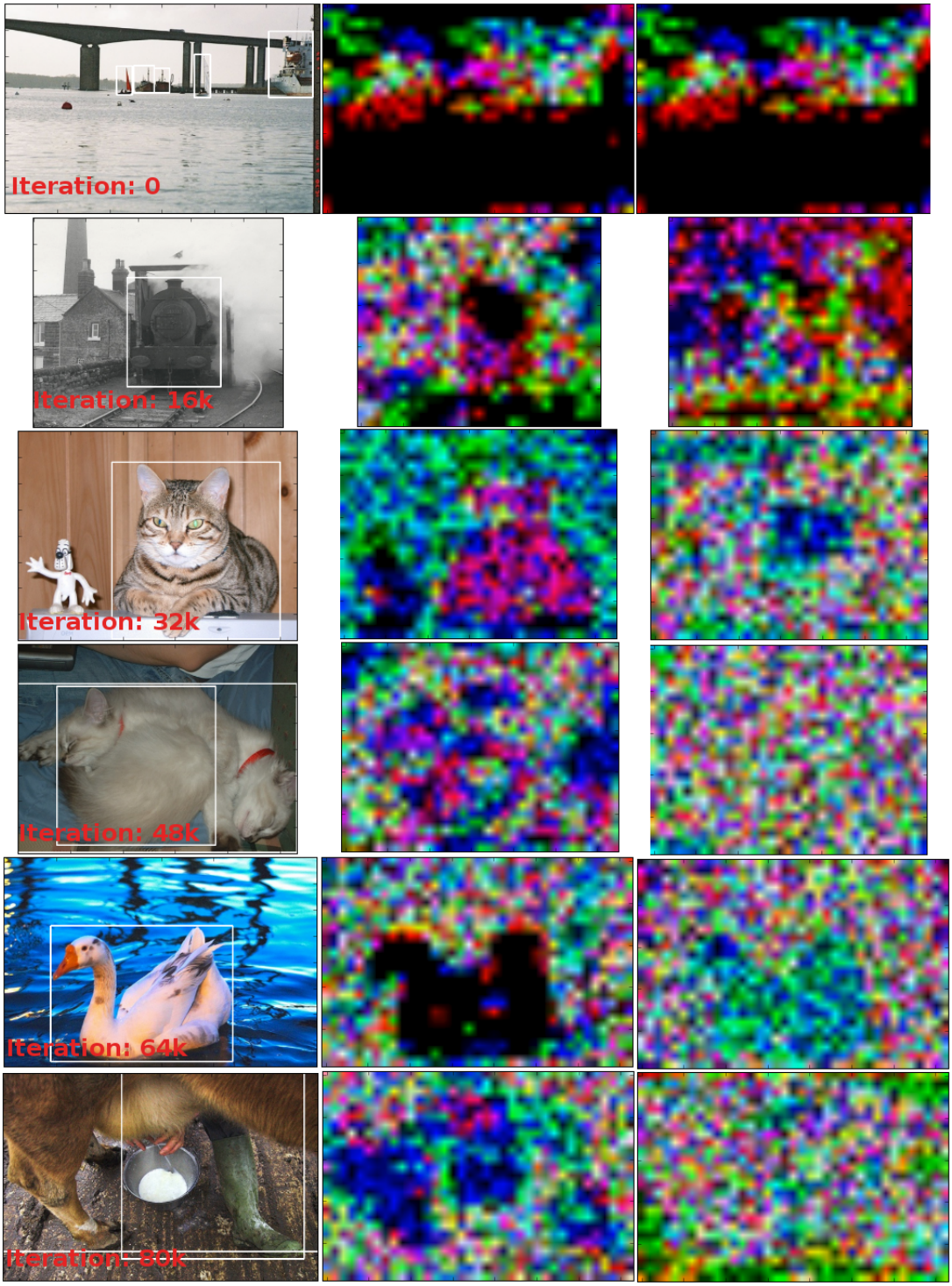

Figure 3.16: Input Image (left), three most activated object features in RGB (middle), three most activated context features in RGB (right).

are also considered similar. False positives that confuse similar classes are defined to be confusion with similar object and vice versa. False positives that do not fit any of the cases described above are called confusion with background.

Distribution of positives for different classes obtained with models B_1 and C_1 are given on Figures 3.17 and 3.18. Horizontal axes of these plots are normalized by the number of instances of the respective class in the test set. On top of the distributions, two different recall curves corresponding to two true positive criteria are plotted. Strong criterion for true positive is the official PASCAL VOC criterion that accepts a detection having IoU higher than 0.5 with a ground truth box as true positive. For weak criterion, the threshold is 0.1. On Figures 3.17 and 3.18; it is observed that in all classes C_1 makes significantly less localization errors than B_1. Also in background errors, there is improvement in most classes. On the other hand, confusion with similar object is slightly increased for certain classes such as person, train and boat. Confusion with other objects is increased for most classes as well. While this increase in confusion type of errors is in general not desired, it should be noted that the increase is mostly in the right portions of the plots where the number of detections made is higher than the number of instances. Therefore, their negative effect on the AP is limited. Based on these results it can be argued that while usage of context information improves localization, sometimes context might also act as a source of confusion.

3.6.2 Sensitivity to Object Characteristics

In this part, we investigate the sensitivity of object detectors to certain visual object characteristics. The characteristics are occlusion, truncation, size, aspect ratio, visible sides and visible parts of the object.

Sensitivity and Impact plots for the models B_1 and C_1 are given in Fig. 3.19. Here, the dashed lines indicate the Average Normalized Precision over all of the test set. Average Normalized Precision is a metric proposed by Hoiem et al. [70]. This metric helps to avoid the imbalance caused by the variation in number of samples over different categories. Class-level effects of factors such as occlusion, bounding box area and visible parts are plotted on Figures 3.25, 3.24, 3.23, 3.22, 3.21 and 3.20.

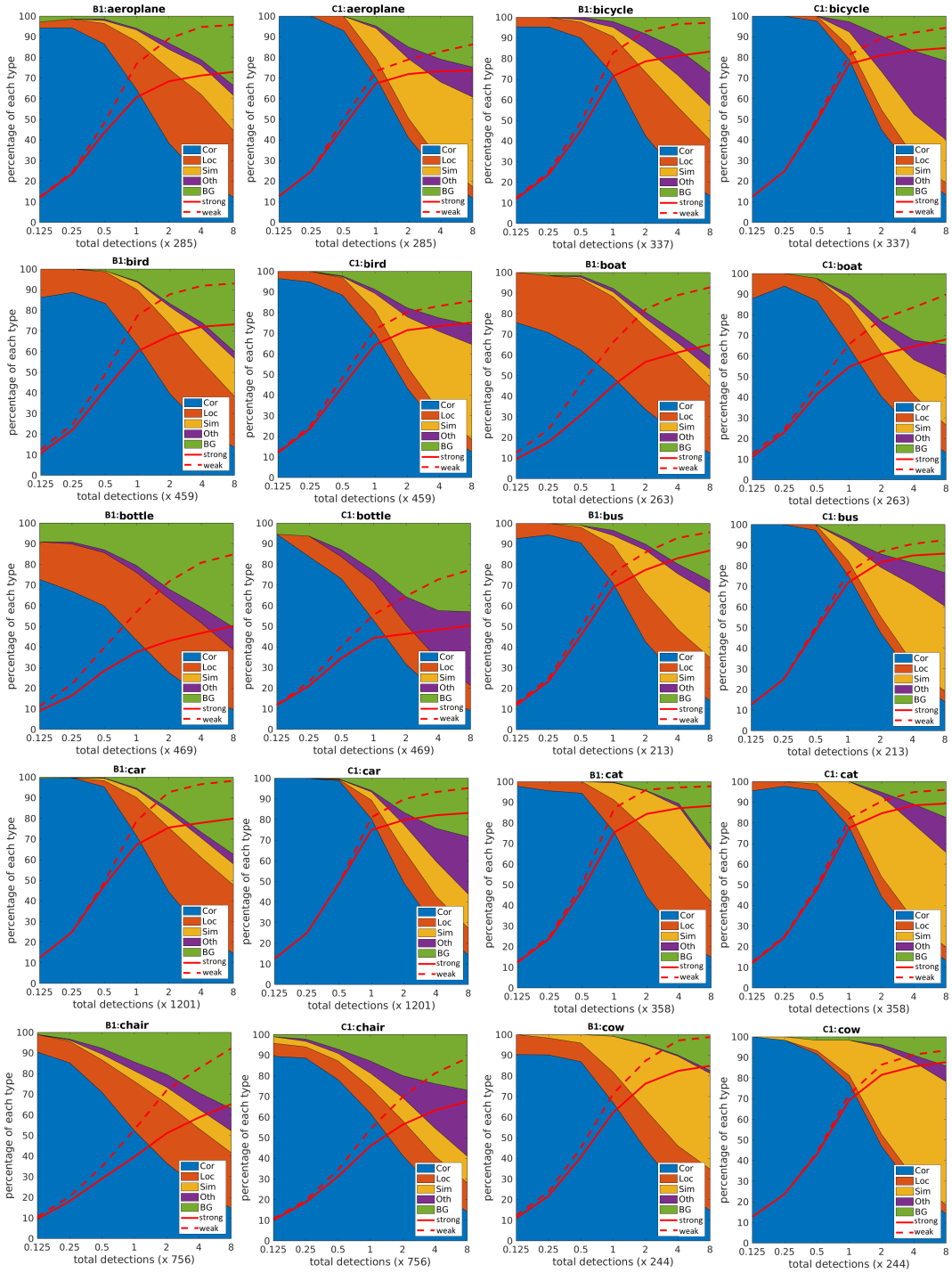

Figure 3.17: Distribution of Positives vs Number of Detections and Recall Curves (Red Lines) for Models B_1 and C_1. Cor: True positives, Loc: Localization errors, Sim: Confusion with similar, Oth: Confusion with other, BG: Confusion with background.

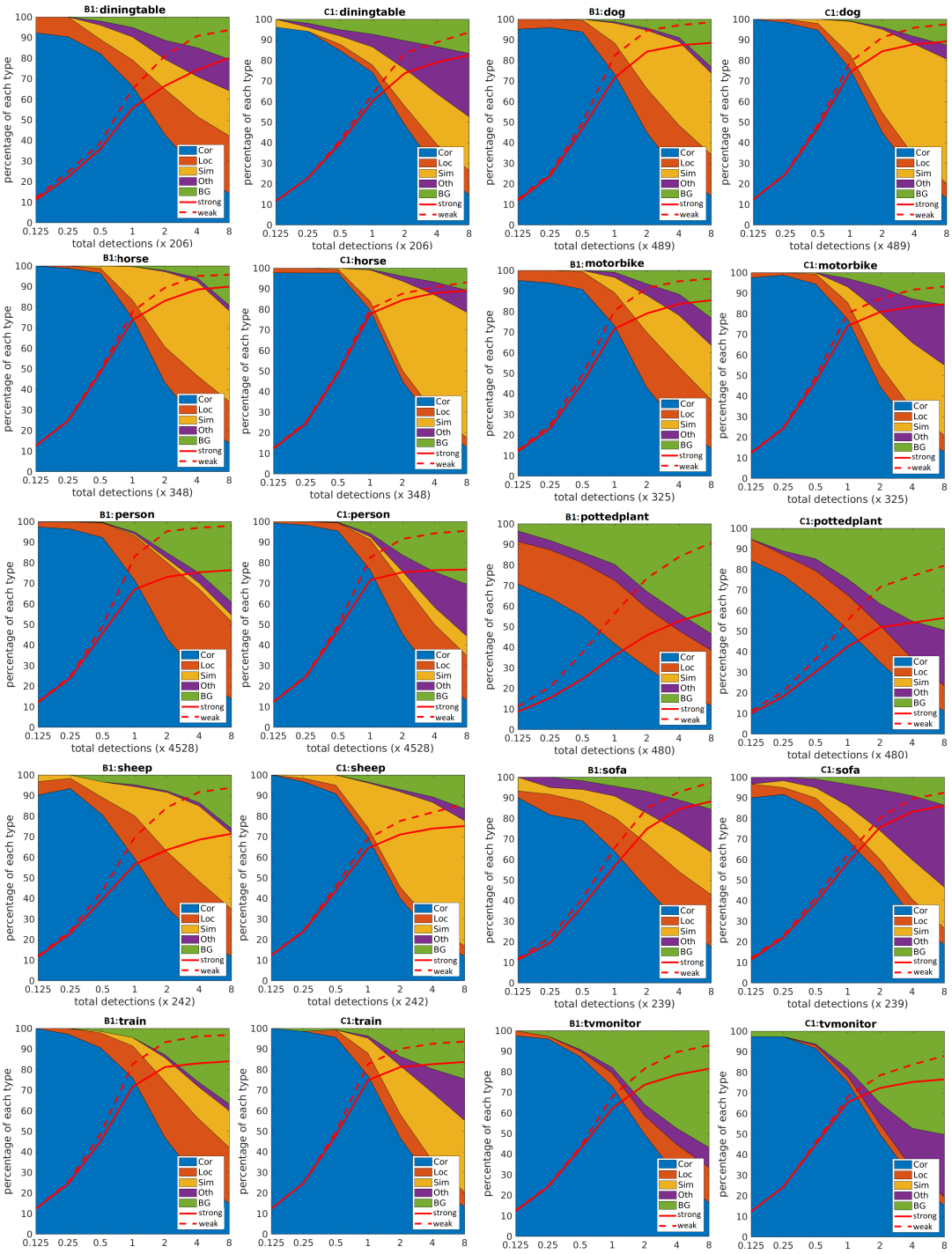

Figure 3.18: Distribution of Positives vs Number of Detections and Recall Curves (Red Lines) for Models B_1 and C_1.

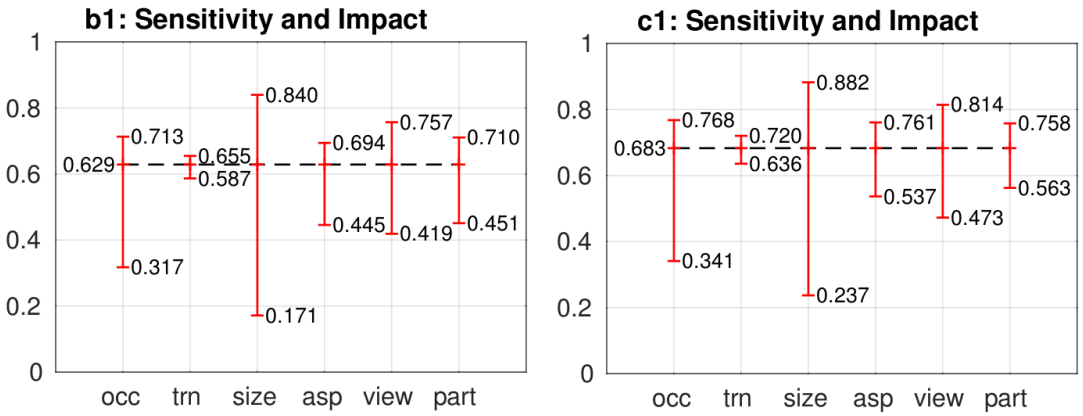

Figure 3.19: Sensitivity (difference between max and min) and Impact (difference between max and overall).

From Figure 3.19, it is observed that there are no drastic differences in the sensitivity and impact between the two models. Moreover, the detailed results in Figures 3.25, 3.24, 3.23, 3.22, 3.21 and 3.20 suggest that the general trends are similar for the baseline and context models. These trends are informative not just about the detectors but also about the properties of the dataset. For both models, occlusion leads to significant drops in performance for classes such as airplane, bird, boat and chair as observed from Figure 3.20. While occlusion is generally considered as a negative factor, it is worth noting that occlusion consistantly increases the detection performance for the table class. This tendency is reasonable since tables are usually occluded in the dataset with instances from related object classes such as bottles and humans. According to Fig.s 3.21, 3.23, as bounding box area and height get larger, detection performance generally increases. Comparing the results for C1 and B1, it can be concluded that usage of context information leads to an improvement without changing the general behavior of the algorithm dramatically.

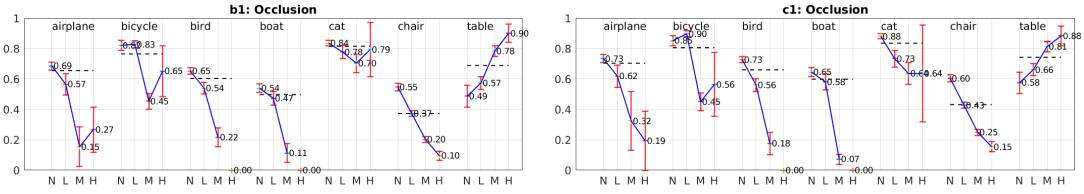

Figure 3.20: Average Precision vs. Level of Occlusion (N: None, L: Low, M: Medium, H: High)

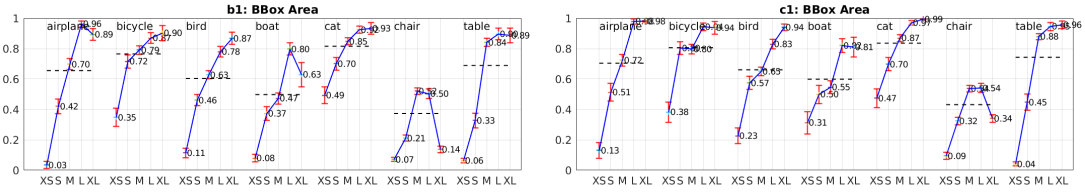

Figure 3.21: Average Precision vs. Bounding Box Area

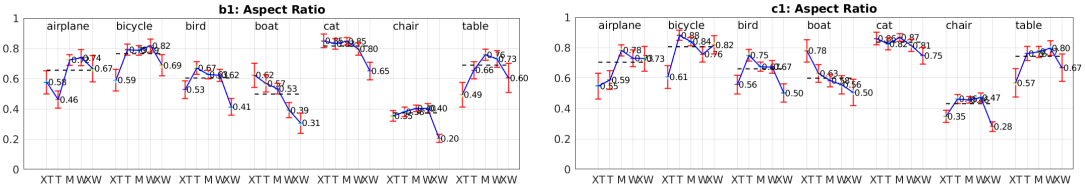

Figure 3.22: Average Precision vs. Aspect Ratio

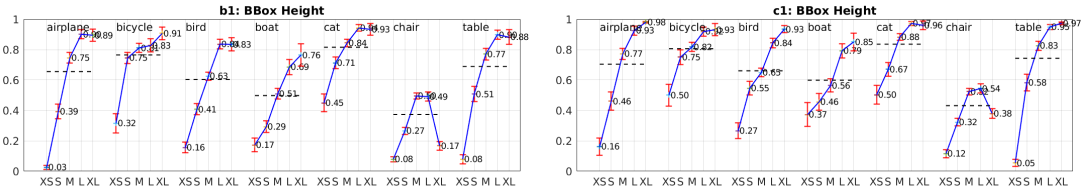

Figure 3.23: Average Precision vs. Height

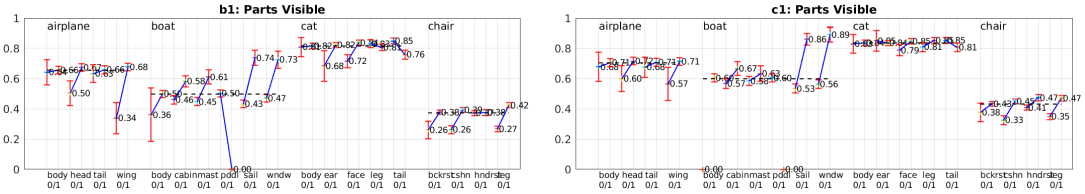

Figure 3.24: Average Precision vs. Parts Visible

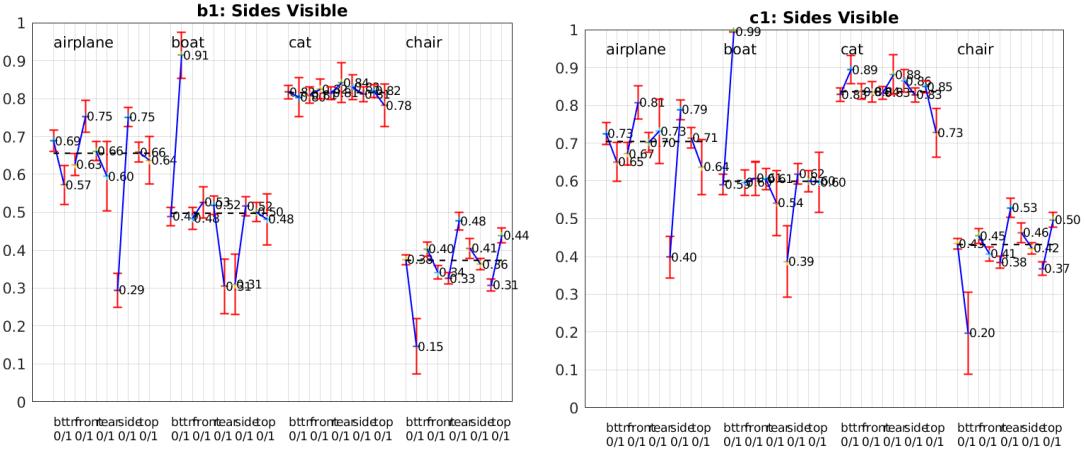

Figure 3.25: Average Precision vs. Sides Visible

CHAPTER 4

VISUAL OBJECT TRACKING LITERATURE

In the most general sense, tracking can be defined as the estimation of the state of a moving object [71]. In visual object tracking, state can be defined as location, deformation, orientation or combinations of these. Estimation of the state can be based on observations related to the object as well as a motion model. Observations, which are sometimes called measurements, can take various forms including radar signals, sonar signals, LIDAR images and videos taken by cameras. Since the early works related with the tracking problem concentrated on military applications, the term "target", instead of "object", was used more often. When the observations are in the form of regular videos, the phrase "visual object tracking" is commonly used in vision literature.

According to the number of objects to be tracked simultaneously, visual object tracking can be divided into single and multiple (visual) object tracking. This thesis focuses on single object tracking. In single object tracking, one is mostly interested in what is referred to as visual tracking of *generic* objects.

4.1 Visual Tracking of a Generic Object

Generic visual object tracking is one of the most commonly studied problems in computer vision. The word generic is used to express the fact that there is minimum constraint on the properties of the tracked object. Object can be an animal, a piece of garbage or even a part of another object (human face). The underlying notion is when designing the generic object tracker, one is not able to make any assumptions regard-

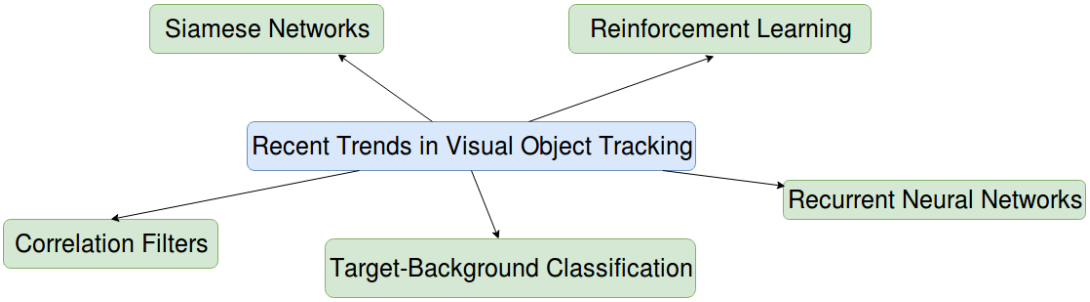

Figure 4.1: Recent Trends in Visual Object Tracking.

ing the shape, appearance or motion of the object. Although, there are many other hand-crafted appearence models in the past, most of the state-of-the-art trackers use deep hierarchical representations (mostly obtained through CNNs) for modeling the visual appearance of objects [72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86].

Object tracking has been studied for long decades and a vast literature has been developed on this topic. In this thesis, we shall narrow down our focus to the most recent developments rather than aiming to draw a complete picture. Specifically, we shall study recent approaches that can be summarized with key concepts illustrated on Figure 4.1.

A respectable number of the many state-of-the-art tracking methods use Discriminative Correlation Filters (DCF) [72, 79, 80, 82, 84, 87, 88, 89]. In DCF-based trackers, in order to obtain an estimate for the location of the object, 2-D spatial correlation between the correlation filter and visual features at each frame is computed in a search window defined around the last location of the object. The filter aims to model the appearance of the object and it undergoes online updates to account for the changes in the appearance of the object. The method MOSSE [90] can be assumed as the pioneer of this group of methods (The term MOSSE is an abbreviation for minimizing the sum of squared error). The error is defined as the difference between the response of a learned correlation filter to input features and desired response. A generic correlation filter based tracking framework is illustrated on Figure 4.2.

As a completely different approach, there are methods that formulate tracking as the problem of obtaining the optimal window out of a set of candidate windows generated at each frame through binary *classification* (i.e. two class: target vs. back-

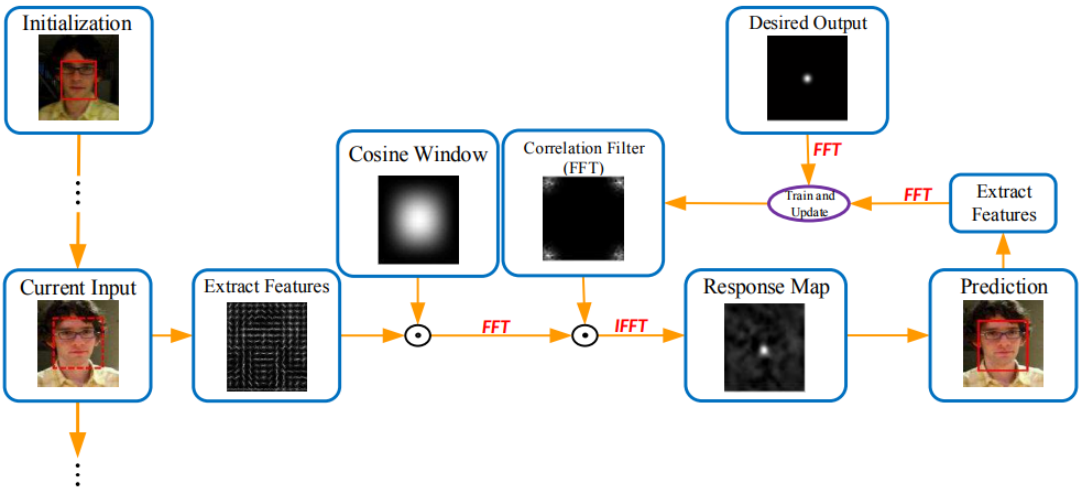

Figure 4.2: A generalized view of correlation filter based tracking. Image taken from: [20]

ground window) [73, 74, 75, 76, 91]. Candidate windows are usually generated from a random distribution centered around the current target window. Target-background classifier can be trained in a supervised manner using the bounding box annotation available in the first frame. Classifier requires to be updated by the information from the incoming frames to handle the changes in target appearance. Tracking based on target-background classification is depicted on Figure 4.3.

It should be noted that both classification-based and correlation filter methods are discriminative: In other words, they aim learning to discriminate between the target and background appearances. Such an approach corresponds to directly estimating a conditional probability of having the target given an observation without attempting to solve the more general problem of estimating the joint probability distribution of having the target and the observation. Accordingly, most of the state-of-the-art trackers adopt a discriminative approach (see Table 5.1).

In order to better match an object in consecutive frames after representing in a different feature space, siamese neural networks are also commonly used in tracking [92, 21, 85, 93, 86, 94]. Siamese networks consist of identical feature extraction stages applied on target, candidate or search patches (patches that are likely to contain target). The general approach is depicted on Figure 4.4. The aim is to learn a similarity metric or a matching function that quantifies how similar a candidate re-

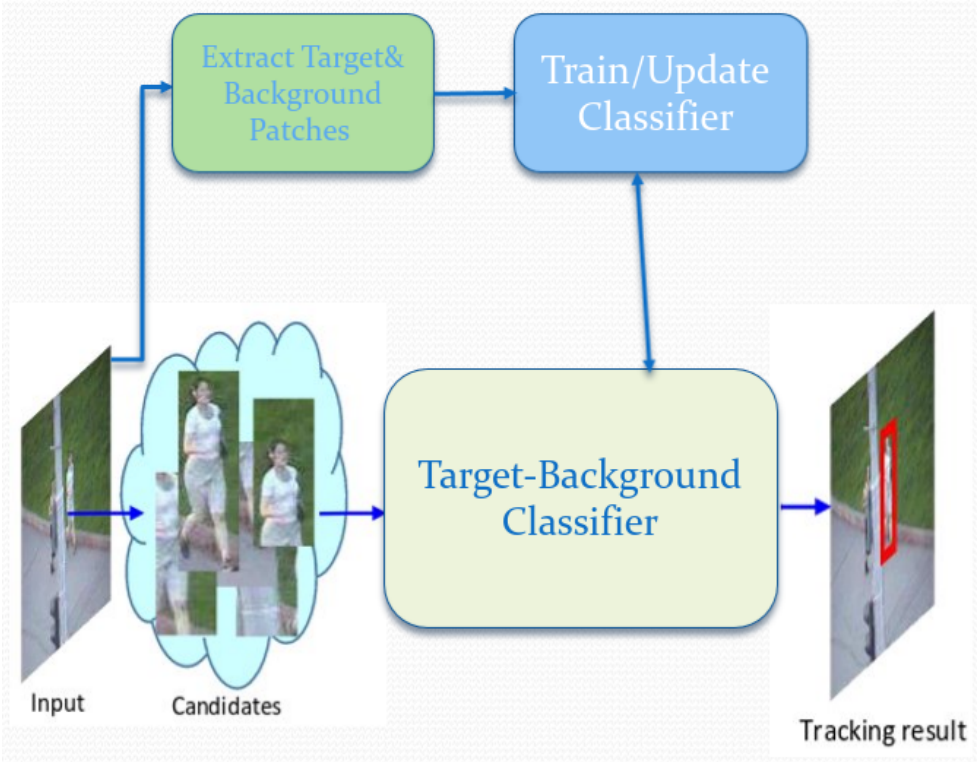

Figure 4.3: Tracking by Target-Background Classification.

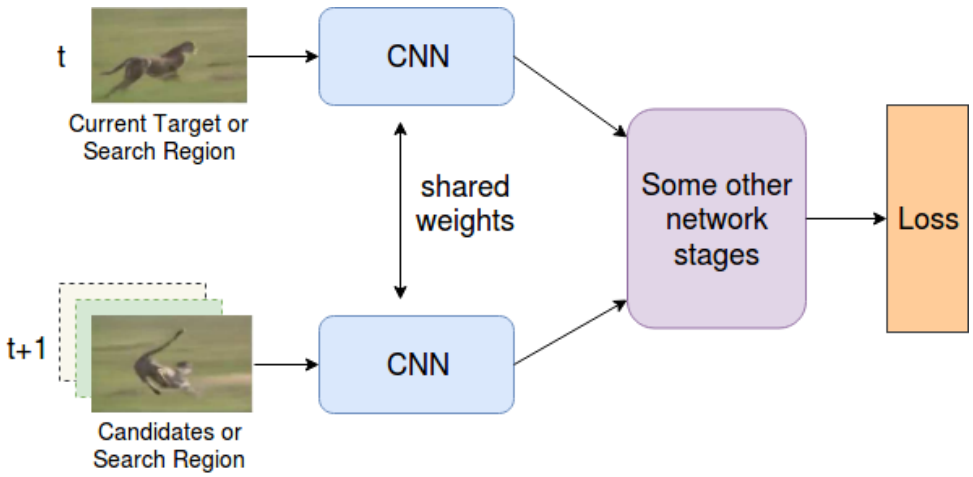

Figure 4.4: Siamese Networks in Object Tracking

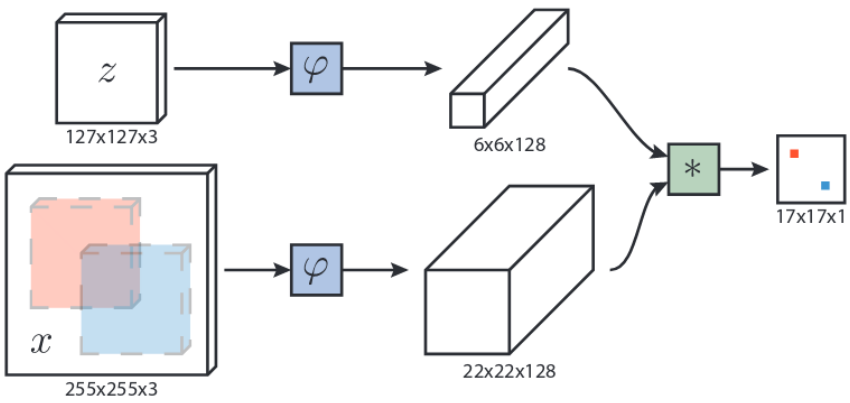

Figure 4.5: Fully-convolutional Siamese architecture proposed in [21]. Feature maps extracted from target (z) and search patch (x) are cross-correlated to obtain a similarity score map.

gion to target is. Seminal work in this branch is SiamFc [21] depicted on Fig. 4.5. In SiamFc, matching of consecutive frame features is performed through correlation.

Some of the recent methods incorporate reinforcement learning [83, 94, 95, 22, 96], which formulates tracking as a problem of policy and decision-making which is a radically different approach. Reinforcement learning is a machine learning paradigm in which a model consists of an agent and an environment. Agent affects the environment by performing an action. As a result of this action, environment undergoes a change that can be sensed through observations. Also, a reward signal is provided by the environment. Agent makes a decision to perform an action based on both the observations and the reward signal. The aim of the agent is to maximize the total reward gathered in the long term.

4.1.1 Visual Object Tracking using Recurrent Neural Networks

Recurrent neural networks have proven to be successful in numerous sequence modelling tasks, such as language modelling [97], image captioning [98] and speech recognition [99]. Object tracking can also be considered as a sequence modelling task. On the other hand, RNN-based tracking models are still underperforming when compared to state-of-the-art correlation filter and classification-based methods mentioned above. RNNs in general maintain an internal state which can encode object

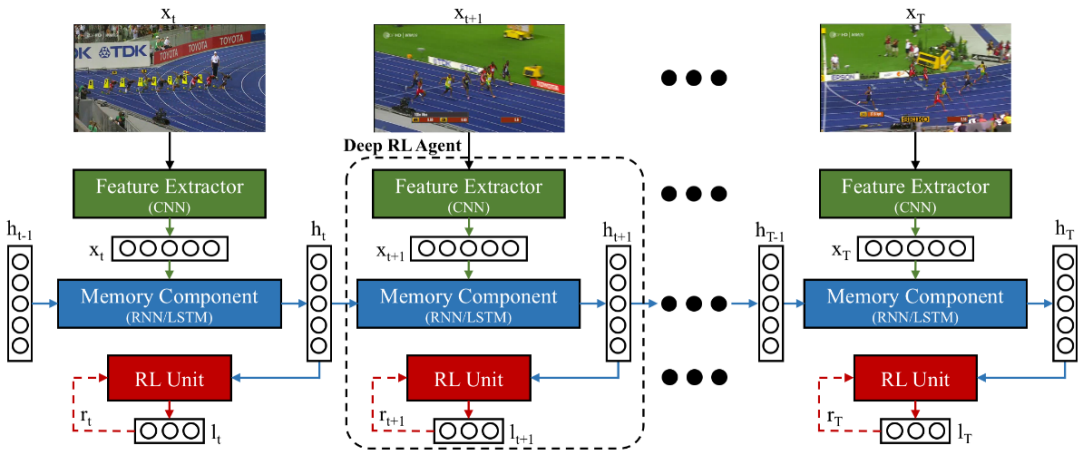

Figure 4.6: DRLT combines Deep Neural Networks with Reinforcement Learning. [22].

motion as well as changes in object appearance. Therefore, an RNN-based tracker network does not need to undergo online parameter updates to handle changes in object appearance. It should be noted that RNNs can be incorporated in a tracking scheme in several different ways. Therefore one should avoid describing a general framework for tracking with RNNs, unlike the case with correlation filters. In this section, we shall review some of the forthcoming RNN-based tracking methods.

Deep Reinforcement Learning Tracker (DRLT) [22], employs RNN as a memory component between a feature extractor and a reinforcement learning (RL) unit. Memory component takes the feature representation as input and maintains a memory state. RL unit decides the target location based on the memory state provided by the memory component. DRLT is depicted on Fig. 4.6.

In Recurrent Attentive Tracking Model (RATM) [23], RNN serves as an attention mechanism that indicates the system the position to examine in the image. The network architecture of RATM is depicted on Fig. 4.7. In Hierarchical Attentive Recurrent Tracking (HART) [24], convolutional appearance features of the object are fed into an LSTM. The output of the LSTM is then fed to a multi-layer perceptron (MLP) which generates attention feedbacks and bounding-box correction. Recurrent Filter Learning (RFL) method proposed by Yang & Chan [25] uses a Convolutional LSTM in conjunction with correlation to locate the target. RFL can be seen as an end-to-end learning based extension of the correlation filter paradigm which employs RNN as a

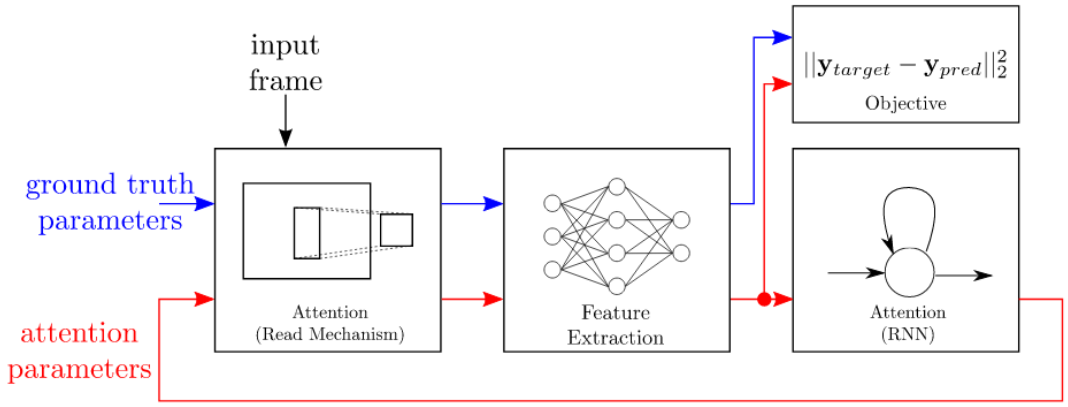

Figure 4.7: Recurrent Attentive Tracking Model [23].

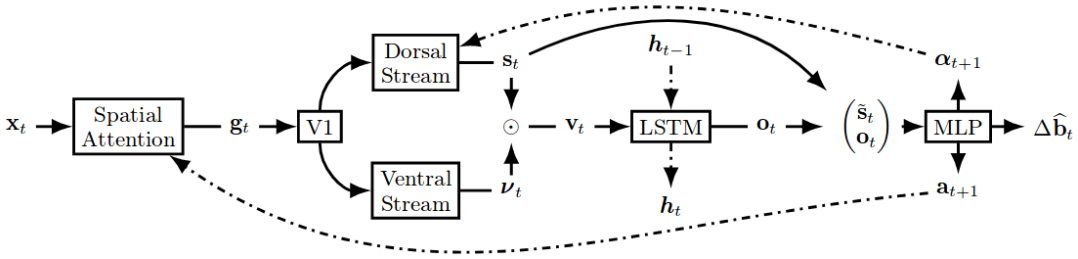

Figure 4.8: Hierarchical Attentive Recurrent Tracking (HART) [24].

trainable correlation filter. RFL is illustrated on Fig. 4.9.

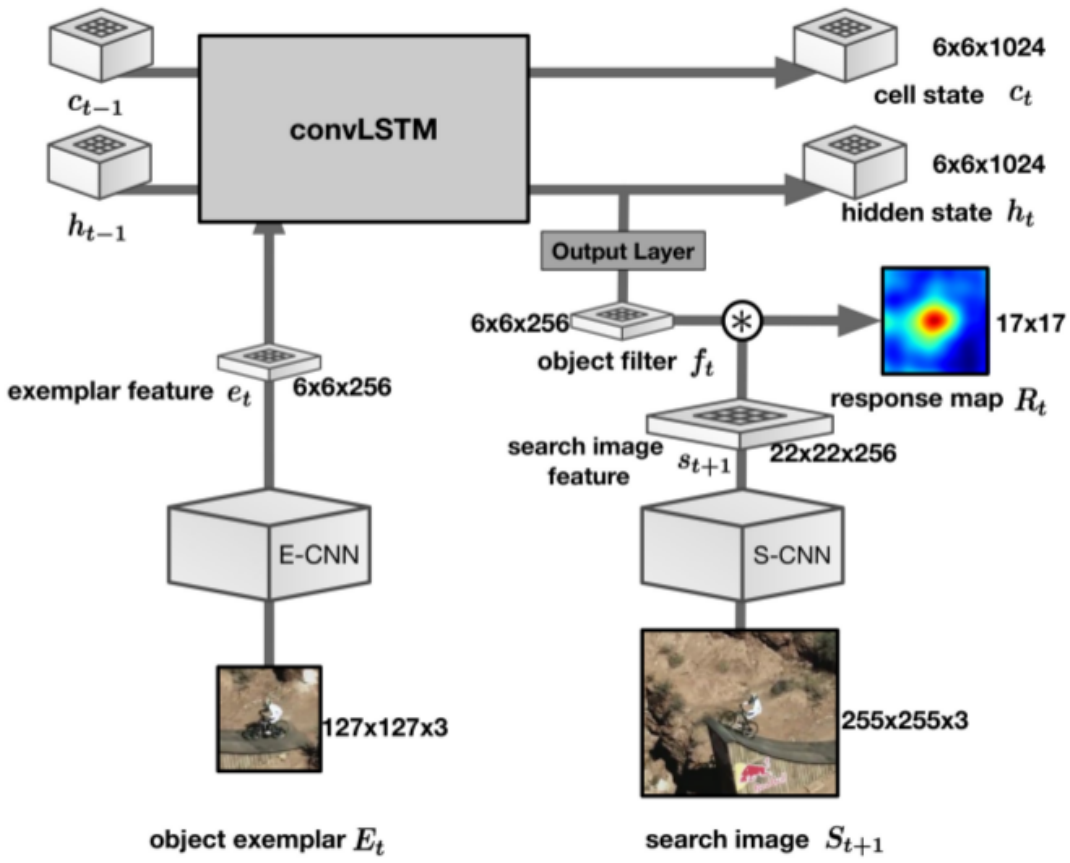

Figure 4.9: Recurrent Filter Learning (RFL) [25].

CHAPTER 5

VISUAL OBJECT TRACKING USING RECURRENT NEURAL NETWORKS

In this chapter, the context feature based object detection method developed in Chapter 3 is combined with an RNN based tracking method, namely Real-Time Recurrent Regression Networks (Re^3) [41], to improve tracking performance. In Section 5.1, Re^3 algorithm is explained in detail. In Section 5.2, a tracking framework that combines Re^3 and object detection with convolutional context features is presented. In Section 5.3, the experiments conducted with Re^3 are presented. Finally, in Section 5.4, experiments with detection-aided tracking are presented.

5.1 Real-Time Recurrent Regression Networks (Re^3)

Re^3 is a method that combines CNN and RNN to track generic objects in real-time. Figure 5.1 illustrates the method for generic object tracking using CNN-based appearance features and RNNs. In this method, at each time step t, the estimated object bounding box is expanded by a certain ratio (by default, doubled). The larger box obtained in this manner is used to crop from both t and $t + 1$ frames. Identical CNN stages are applied to obtain CNN features from both crops. For clarity, certain parts of this system is denoted as Block A, B and C; the experiments are described by refering to these building blocks. In this convention, feature extractor step is denoted as Block A. The convolutional feature maps are concatenated and fed to a fully-connected layer (Block B) followed by the RNN stages (Block C). The output of the RNN stages which is equal to the hidden state is fed to a fully-connected layer

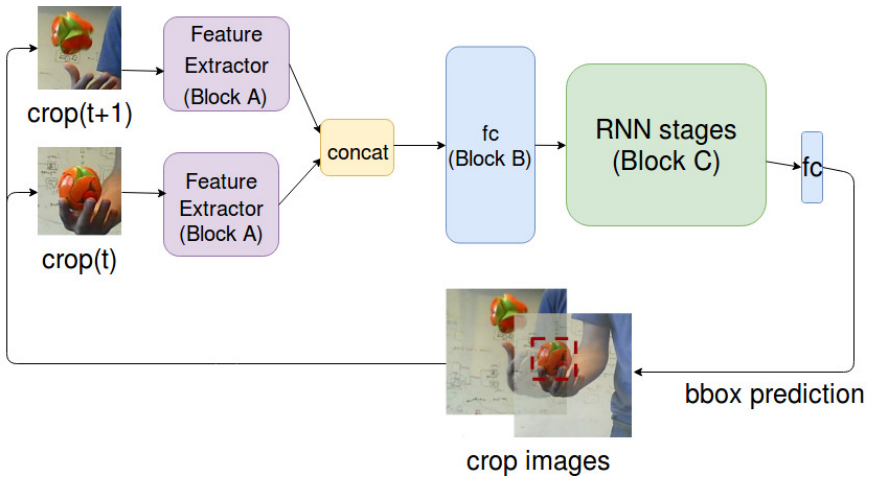

Figure 5.1: Re^3 Network Structure.

with 4 outputs to generate a new estimate for the position in the form of a bounding box. The whole system is trained end-to-end. L_1-loss for bounding box regression and L_2 weight regularization is applied during training.

Bounding box regression loss is expressed as;

$$L_{box} = \frac{1}{ebs} \sum\nolimits_{f \in (frames)} \sum\nolimits_{c \in (x_1, y_1, x_2, y_2)} |c^f_{gt} - c^f_{pred}| \tag{5.1}$$

where, ebs is the effective batch size (number of predictions) which can be computed as batch size times (number of unrolls + 1).

Regularization term is expressed as;

$$L_{reg} = \lambda_{reg} \sum\nolimits_{w \in net\ params} w^2 \tag{5.2}$$

where, $netparams$ is the set of trainable parameters in the network.

Total Loss is defined in Eqn. 5.3.

$$L_{total} = L_{box} + L_{reg} \tag{5.3}$$

In the original work [41], Long Short-Term Memory (LSTM) [100] is used in the RNN part. In the thesis, the results with LSTM, as well as its popular variant Gated

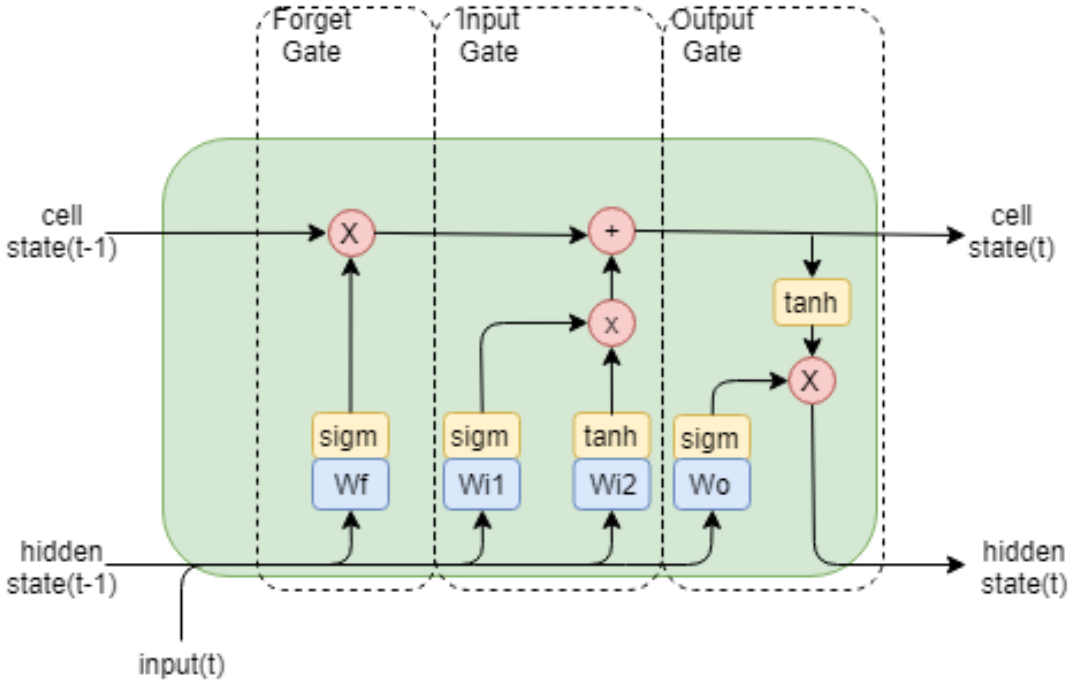

Figure 5.2: Long Short-Term Memory.

Recurrent Unit (GRU) [101] and Convolutional LSTM are presented. A brief overview of these different types of RNNs are presented next.

LSTM architecture is depicted on Fig. 5.2. LSTM cell has two types of states called hidden and cell states. Inside an LSTM cell, nonlinearities play an important role in bringing numbers to a desired range so that the resulting system is numerically stable. Sigmoid (σ) and $tanh$ function have ranges [0,1] and [-1,1], respectively. At each time step, sequential input is concatenated with the hidden state and the resulting combined information is processed by three gates consisting of linear weight layers and nonlinearities. For each element in the cell state vector, forget gate generates a value between 0 and 1 that expresses the importance of the corresponding cell state element. Some of the information accumulated in the cell state is forgotten in this manner and the decision to forget is performed based on the values of both the current input and the current hidden state. The next gate is denoted as the input gate and it generates values between 0 and 1 to decide how much of the incoming information will be stored in the cell state. The final gate is defined as the output gate and it is responsible for generating the next hidden state.

A well-known variant of LSTM is the one with peephole connections [102]. In this

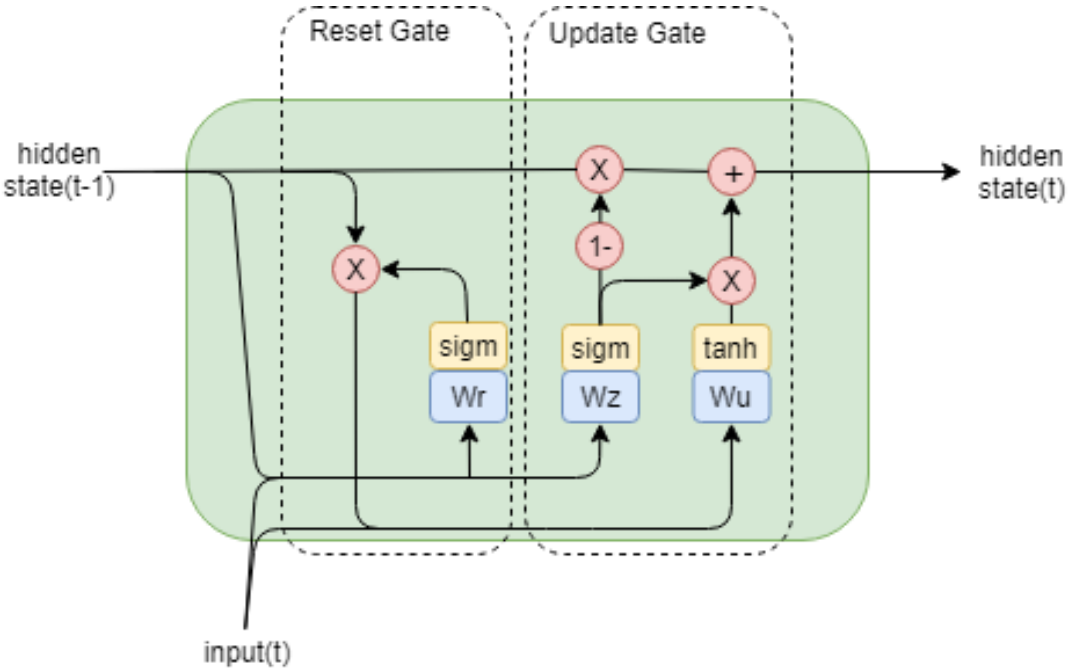

Figure 5.3: Gated Recurrent Unit.

version; forget, input and output gates are fed not just the hidden state and the input but also the cell state is fed. Re^3 algorithm uses this version of the LSTM.

Gated Recurrent Unit depicted on Figure 5.3 has a single state vector as the hidden state. There is a reset gate and an update gate. Reset gate decides on how much of the information stored currently in the state will be preserved whereas, update gate decides on how much of the incoming information will contribute to the state in the next timestep.

Considering training of the recurrent networks, it should be noted that feed-forward neural networks (including CNNs) are usually trained by applying the backpropagation algorithm [103]. Backpropagation is a systematic application of the chain rule that computes the gradient of the cost function with respect to every trainable parameter in the network in a computationally efficient manner. Computational efficiency is achieved by propagating an error term among consecutive network layers rather than computing everything from scratch at each layer. Backpropagation Through Time (BPTT) [104], extends backpropagation to dynamic structures such as RNNs. In BPTT, a RNN is unfolded along time axis as depicted on Figure 5.4 so that it is reformulated as a feed-forward network. In this new formulation, identical copies

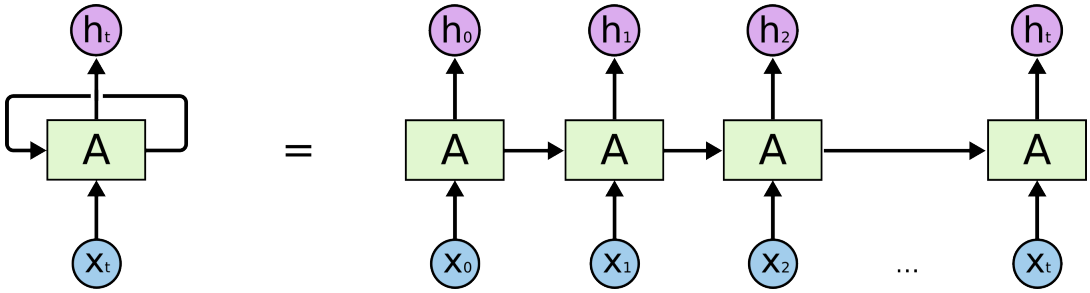

Figure 5.4: Unrolling of an RNN cell [26].

of the same cell are present for each timestep and the state is passed between them through feed-forward connections in time. The sum of all time step costs is taken to obtain a total cost and the gradient of this total cost is used to update the parameters. Since the parameters are shared in each consecutive copy, gradients computed for different timesteps are taken average to update parameters. It is possible to perform unfolding along the whole sequence, although it is often considered impractical during training. Therefore, truncated BPTT is frequently preferred. In this method, BPTT is performed in limited number of unrolls.

Training is performed by applying ADAM optimization [105]. ADAM is a gradient descent type of optimization algorithm. Gradient descent computes the gradient of the loss function with respect to parameters of a system using the whole training set. Stochastic Gradient Descent (SGD), estimates the gradient using randomly constructed mini-batches. SGD directly uses gradients estimated from mini-batches to update parameters. Unlike SGD, ADAM optimizer uses first and second moment estimates of the gradients that are obtained through exponential moving averages to update parameters [105].

5.2 Combining Tracking and Detection

In this section, we combine the context-based detection algorithm proposed in Chapter 3 with the Re^3 model and build a detection-aided tracking model which is depicted on Figure 5.5.

In this framework, tracking starts by feeding the initial frame and the corresponding ground truth bounding box to the tracker network. At every 5 frames, *Check Failure*

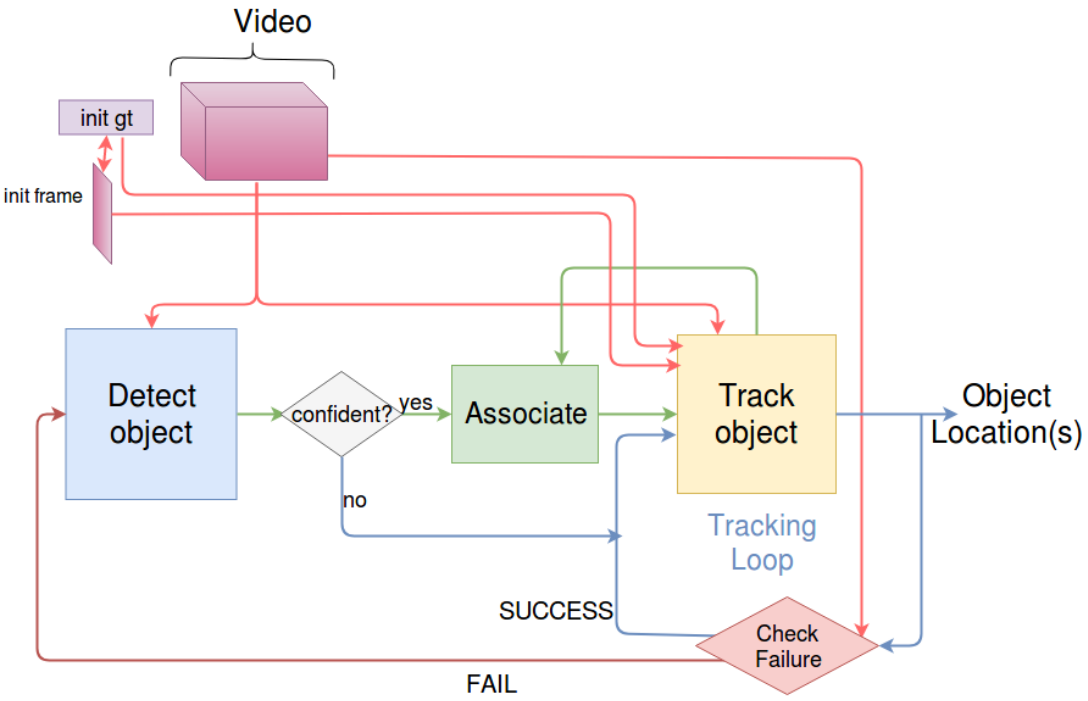

Figure 5.5: Detection-aided tracking model.

block checks whether the tracker fails to track the object.

Failure check is performed as follows: Tracker output is fed to the detection network as a proposal. Classifier output of the detection network is used to decide whether the tracking has failed or succeeded. If the class confidence of the true class of the object is below a certain threshold, namely *failure threshold*, it is regarded a failure. It should be noted that this is an *estimated failure*, not necessarily an actual failure. The failure threshold is a hyperparameter that determines the amount of the classification result to be relied upon. *Check Failure* blocks constructed from context detector (C_1) and baseline detector (B_1) are depicted on Figure 5.6.

Whenever the *check failure* block estimates that the tracking has failed, single image detection is performed by directly applying the detection algorithm (B_1 or C_1). Detection results are confidence thresholded so that the detections that surpass the user-defined confidence threshold are defined as confident detections. Detector confidence threshold is a hyperparameter that reflects the amount the detector is relied upon. If there are no confident detections, tracking continues in the usual manner. Otherwise, association between the detections and the track is performed by comput-

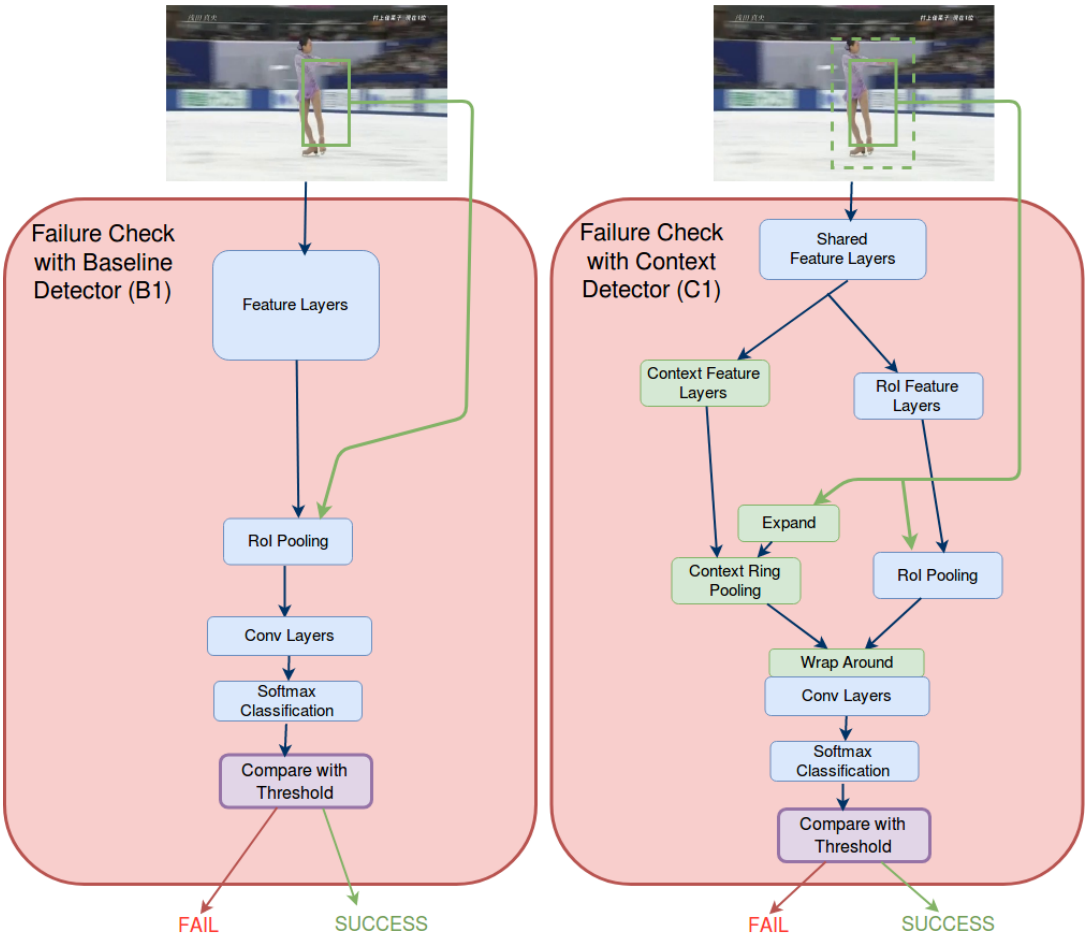

Figure 5.6: Checking failure of a track with baseline detector (left) and context detector (right).

ing L2 distances between the most recent track box and the confident detection boxes. Detection that is closest to the last track box is used to reinitialize tracking.

During reinitialization, RNN states are also reset. It is worth noting that failure checking performs classification on a single bounding box which is computationally not costly, whereas detection step proposes many regions and classifies and regresses all of them which constitute a significant computational cost.

5.3 Re^3 Experiments

In this section, we describe the experiments performed with Re^3. In these experiments, we investigate the effect of several design choices on the performance of a Re^3 model. In deep neural network models, theoretical justification is limited when making design choices. Therefore, controlled experiments play an important role in the design process. Section 5.3.1 describes the training data used in the experiments. Section 5.3.2 describes the evaluation procedure. Section 5.3.3 describes the experiments and their results.

5.3.1 Training Data

It is important to realize that since Re^3 is a deep neural network model, the success of the method depends heavily on the amount of training data. The training is performed on the Imagenet VID [40] dataset. Imagenet VID contains 3,862 training videos with a total of 1,122,397 frames and 7,911 unique object tracks featuring instances from 30 animal and vehicle categories. VID categories are; airplane, antelope, bear, bicycle, bird, bus, car, cattle, dog, domestic cat, elephant, fox, giant panda, hamster, horse, lion, lizard, monkey, motorcycle, rabbit, red panda, sheep, snake, squirrel, tiger, train, turtle, watercraft, whale and zebra.

5.3.2 Evaluation

We evaluate the performance of the algorithm on the VOT2016 dataset. VOT2016 consists of 60 sequences of daily objects. In VOT challenge [106], there are three primary evaluation metrics: Accuracy, Robustness and Expected Average Overlap (EAO). These metrics are computed in a supervised framework. In this framework, when the tracker completely loses the target (zero overlap), it is called a failure and the tracker is reinitialized with the ground truth five frames later. In order to eliminate possible bias, ten frames after reinitialization are ignored when calculating accuracies.

Accuracy is quantified with the average overlap between ground truth and predictions over all frames. Formally speaking, let the overlap (ϕ_t) of the ground truth region at time t (R_t^G) and tracker output at time t (R_t^T) be expressed as follows:

$$\phi_t = \frac{R_t^G \cap R_t^T}{R_t^G \cup R_t^T} \tag{5.4}$$

Accuracy of the tracker is,

$$\bar{\phi} = \sum\nolimits_t \frac{\phi_t}{N} \tag{5.5}$$

Robustness indicates whether the tracker fails too many times or not. Robustness is defined as,

$$R_S = e^{-SM} \tag{5.6}$$

where M denotes the mean-time-between-failures and S is by default set to 30. Accuracy and Robustness are usually reported together. It is commonly encountered that one algorithm is better in terms of accuracy whereas the other algorithm is better in terms of robustness when comparing the performance of two algorithms. Tracker performance is high if both accuracy and robustness are high [107]. Number of failures is also an indicator for robustness. Therefore, instead of computing robustness using Eqn. 5.6, here we frequently report number of failures.

Expected Average Overlap (EAO) metric was introduced in VOT 2015 [108]. EAO aims to unify both the accuracy and robustness performances of a tracker in a single value. For a sequence having length N_s, average overlap $\phi(N_s)$ is computed by averaging the prediction and ground truth overlaps at each frame. $\widehat{\phi(N_s)}$, which is the expected average overlap for a sequence length N_s, is estimated by evaluating the average overlap for a large number of sequences having the same length. An expected average overlap curve is obtained by evaluating expected average overlap at different sequence lengths. Average of this curve over an interval of typical lengths is reported as the EAO value. Formally speaking, let ϕ_i be the per-frame overlap at the i'th frame of an N_s frames long sequence, expected average overlap curve is defined as;

$$\widehat{\phi(N_s)} = E[\phi(N_s)] = E[\frac{1}{N_s}\sum_{i=1}^{N_s}\phi_i] \tag{5.7}$$

EAO is computed from this curve as follows:

$$EAO = \frac{1}{N_{high} - N_{low}}\sum_{N_s=N_{low}}^{N_{high}}\widehat{\phi(N_s)} \tag{5.8}$$

N_{high} and N_{low} are higher and lower limits for typical video sequences. EAO of state-of-the-art trackers on VOT 2016 are presented on Table 5.1. Trackers are grouped into types as Target-Background Classifier (TB), Correlation Filter (CF) and other approaches.

In addition to accuracy and robustness, speed of a tracker is also important. Speed is difficult to measure since it depends on several factors including hardware, programming language whatsoever. Even execution of the same algorithm twice on the same hardware should not yield the exactly the same execution time due to numerous physical factors. Execution time is measured in terms of frames-per-second (FPS).

5.3.3 Experiments

In order to observe the effects of several design choices on the operation of the system, we train models under different configurations.

Table 5.1: Expected Average Overlap (EAO) of state-of-the-art trackers on VOT 2016.

Tracker	**Type**	**EAO**
CFCF [84]	CF	0.390
ECO [72]	CF	0.374
C-COT [88]	CF	0.331
TCNN [76]	TB	0.325
SSAT (a version of MDNet [73])	TB	0.321
MLDF [109]	TB	0.311
Staple [89]	CF	0.295
DDC [109]	CF	0.293
EBT [91]	TB	0.291
SRBT [109]	Other	0.290

In all experiments, ADAM optimization [105] is applied with a learning rate of 10^{-5} in the first 10k iterations and 10^{-6} in the rest of the training. λ_{reg} of weight regularization term in Eqn. 5.2 is set 0.0005.

In the original work [41], test time RNN reset length is determined according to training unrolls. For example, a model trained with a maximum unroll of 32, RNN states are reset every 32 frames. It is assumed that RNN might not be able to generalize to unobserved sequence lengths. We follow the same strategy in Experiments 1-7, so that every test in these experiments are performed by resetting RNN states at every N frames, N being the maximum number of unrolls made when training the model at stake. In Experiment 8, we test the same model with different state reset periods.

Default feature extraction backbone network is the well-known CaffeNet variant of Alexnet [57] and it is initialized with Imagenet pretrained weights. Backbone network is depicted on Figure 5.7. The network involves convolutional stages each followed by a ReLU nonlinearity. Local Response Normalization (LRN) is applied after the first two convolution layers. LRN normalizes the feature maps across different channels that is observed to be helpful for generalization [57]. During experiments, except Experiment 2, default feature extraction network is employed. In Experiment 2, different feature extractors are compared.

Figure 5.7: Caffenet feature extraction backbone.

Table 5.2: Training schedule with varying number of unrolls

iters	0-20k	20k-40k	40k-60k	60k-80k	80k-100k
Unrolls	2	4	8	16	32
Batch Size	32	16	8	4	2

5.3.3.1 Experiment 1: Effect of Unrolls During Training

In order to observe the effect of number of unrolls applied during training to model performance, we train a series of models applying different unroll strategies. For a fair comparison, model architecture in these experiments is kept the same. Block A is the Caffenet backbone which is depicted on Figure 5.7. Block B is a fully-connected layer with 1024 units. Block C is a single LSTM stage with state sizes being 512. We train 3 models for 100k iterations.

Model V5 is trained with the regularly increasing number of unrolls strategy proposed in [41]. In this strategy, training begins with 2 unrolls and a mini-batch size of 32. at each 20K iterations, the number of unrolls is doubled while dividing batch size by 2 so that the effective batch size remains the same (64) throughout the whole training. By keeping the effective batch size the same, it is ensured that same amount of computation capacity is used as the other 2 models for a fair comparison. Unroll and batch size schedule of Model V5 is summarized on Table 5.2. Model C16 is trained with constant 16 unrolls and a batch size of 4. Model C32 is trained with constant 32 unrolls and a batch size of 2. Model C2 is trained with constant 2 unrolls and a batch size of 32.

The loss curve behaves significantly different than the case of having a constant number of unrolls when training with varying number of unrolls. The bounding box regression losses and unrolls vs. iteration for models V5 and C16 are depicted on Figure 5.8. Loss makes a sudden jump and it can never reach back to the lower values achieved with less number of unrolls whenever the number of unrolls is doubled.

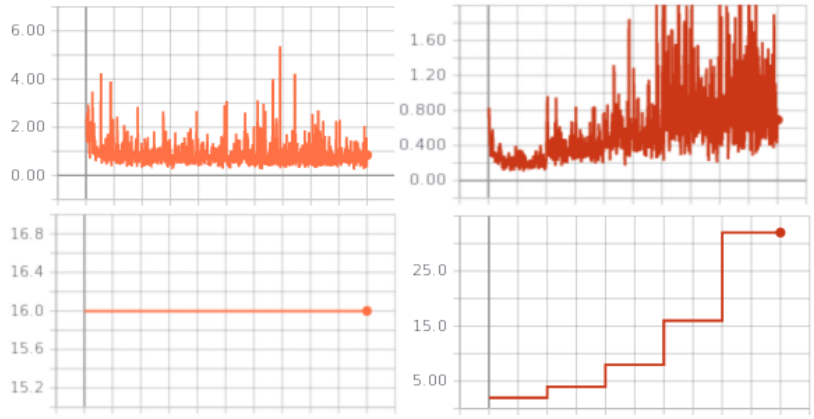

Figure 5.8: Loss Curves (top) and Number of Unrolls (bottom) vs. Iteration plots for models C16 (left) and V5 (right). 10k iterations between vertical lines.

Table 5.3: Results for Experiment 1: Effect of Unrolls During Training

Model Name	**Unrolls**	**Accuracy**	**Failures**	**EAO**	**Speed (FPS)**
V5	2-4-8-16-32	0.51	**24**	**0.18**	38.92
C2	Constant 2	0.36	31	0.14	32.80
C16	Constant 16	0.51	**24**	**0.18**	38.42
C32	Constant 32	**0.54**	25	0.16	**39.11**

This behavior might be reasonable in the sense that with a new unroll number, we are actually forcing the network to learn a new and more complicated task: Instead of tracking an object for N frames, it is now required to track it for $2N$ frames. Number of frames might not seem to make an important difference when thinking about human visual perception. Nevertheless experimental results reveal that it makes an important difference when training RNNs.

Results for models V5, C2, C16 and C32 are presented on Table 5.3. According to these results, accuracy-robustness performance of V5 and C16 are measured to be equivalent. On the other hand, C32 which was trained with 32 unrolls has better accuracy but is slightly less robust than the other two models. In EAO metric, V5 and C16 are better than the other models. These results suggests that there is not a significant difference between training with varying unrolls versus with a constant unroll length that is properly selected. On the other hand, if constant unroll length is selected to be too small or too high, as in the cases of model C2 and model C32, performance gets significantly worse. System performance is quite sensitive to this

hyperparameter. Therefore, in order to eliminate the need to determine the optimal constant length, training with varying unrolls is a reasonable option.

5.3.3.2 Experiment 2: Effect of Number of Feature Extraction Stages

As a consequence of the hierarchical nature of CNNs, consecutive feature layers in a CNN correspond to various levels of abstraction. While the first layer represents low level visual properties such as corners and edges, the last feature layer carries semantic information and there is a gradual transition in between. In certain computer vision tasks, such as image recognition and semantic segmentation, quite deep CNN architectures have proved to be successful. On the other hand, most of the successful CNN-based tracking methods are based on shallow CNNs [73, 74, 75, 76, 77] which suggests that quite deep representations might not be desirable when tracking objects. Execution time consideration might also play a crucial role in making this decision.

With this motivation, controlled experiments with varying number of feature layers are performed to investigate the effect of feature extraction depth. Specifically, we train three models called F1, F3 and F5 which are having 1, 3 and 5 conv layers, respectively, as indicated on Table 5.4. In Model F5, Block A is the default feature extractor which is depicted on Fig. 5.7. Block A's of F1 and F3 are depicted on Fig. 5.9.

All three models are trained for 80k iterations with a constant unroll length of 16 and a batch size of 4. Block B is a fully-connected layer of size 1024 and Block C is a single layer LSTM stage with 512 element state vectors.

Quantitative Results for models F1, F3 and F5 are presented on Table 5.4. According to these results, model F5 is the best model in terms of accuracy and robustness. On the other hand, F1 is the best model in terms of speed. This results suggest that as the number of feature layers increase, both accuracy and robustness of the model increases significantly. Since number of layers is directly related with computational complexity, speed of the tracker decreases as the number of layers increase.

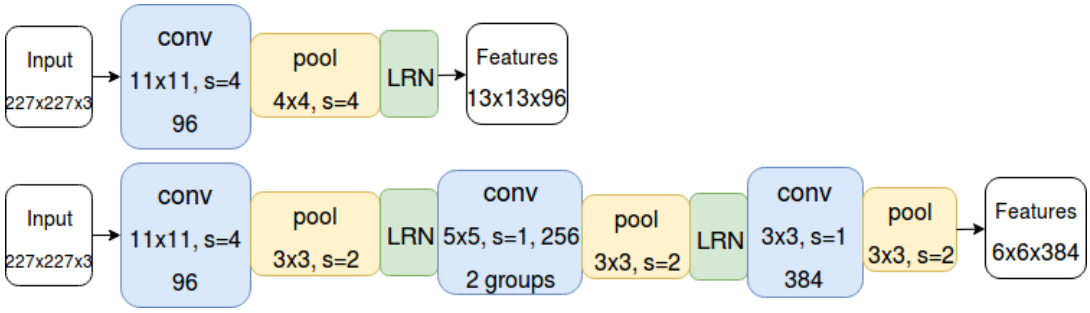

Figure 5.9: 1-layer and 3-layer feature extractors of Models F1 and F3 in Experiment 2.

Table 5.4: Results for Experiment 2: Effect of Number of Feature Extraction Stages

Model Name	Num Convs	Accuracy	Failures	EAO	Speed (FPS)
F1	1	0.35	59	0.09	**44.44**
F3	3	0.49	45	0.14	40.24
F5	5	**0.52**	**25**	**0.16**	38.73

5.3.3.3 Experiment 3: Effect of the size of the two-frame representation

In Re^3, CNN features are passed into a fully connected layer denoted Block B on Figure 5.1. Block B produces a two-frame representation that aims to represent the essential information within two consecutive image crops useful for matching them. There is no cookbook for deciding the size of this representation and all we can achieve is to perform controlled experiments for determining a decent size. Therefore, models with different two-frame representation sizes are trained and compared in terms of their performance. All models are trained for 60k iterations.

Model feature sizes and quantitative results for models R1, R2 and R3 are presented on Table 5.5. Based on these results, it can be concluded that size of Block B is not that critical for the performance of the algorithm. While the model that has the least number of neurons is the one that is most accurate, EAO turn out to be equal for all three models. Speed advantage of R1 makes it more favorable.

Table 5.5: Results for Experiment 3: Effect of Feature Vector Size

Model	Feat. Neurons	Accuracy	Failures	EAO	FPS
R1	256	**0.53**	32	**0.16**	**40.08**
R2	1024	0.52	**29**	**0.16**	38.44
R3	2048	0.51	34	**0.16**	36.25

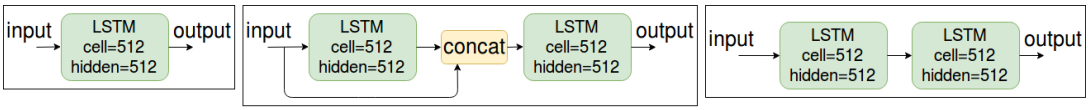

Figure 5.10: Block C of Models S1 (left), SC2 (middle) and S2 (right).

5.3.3.4 Experiment 4: RNN stages

While a single stage RNN seems to be sufficient for modeling a sequence, multi stage RNNs are also frequently used. In order to investigate the effect of this difference, we train three models having single stage and two stage Re^3 models. Block C of Model S1 is a single stage LSTM. Block C of Model SC2 is a two stage LSTM. Block C of Model S2 also is a two stage LSTM but this time, only the output vector of the first LSTM stage is fed to the second stage. By comparing SC2 and S2, one might decide whether it is necessary to feed fully connected features to the second stage. Block Cs of models in this experiment are depicted on Figure 5.10.

Quantitative Results for models S1, SC2 and S2 are presented on Table 5.6. According to these, best configuration for the RNN stages is two stage with the short connection that feeds fully-connected features to the second LSTM stage. Most accurate model is the one with 2 stage LSTM and no short connection. This is also the least robust model.

5.3.3.5 Experiment 5: Evolution of Performance During Training

Training neural networks take quite long time, especially for vision tasks; hence, efficient usage of time is an important concern in vision research. Moreover, in order to gain insight into the learning process, it is important to measure the performance of the network at different iterations. We evaluate the performance of two different models at each 20k iterations. Training CU is performed with a constant unroll of

Table 5.6: Results for Experiment 4: RNN stages

Model Name	RNN Stages	Accuracy	Failures	EAO	FPS
S1	1-stage	0.51	24	0.18	38.92
SC2	2-stage with concat	0.50	**23**	**0.19**	**39.15**
S2	2-stage no concat	**0.52**	33	0.18	37.27

Table 5.7: Results for Experiment 5, Model VU

Model Name	Unrolls	Accuracy	Failures	EAO	Speed (FPS)
VU20	2	0.44	50	0.12	33.65
VU40	2-4	0.48	32	0.16	36.76
VU60	2-4-8	0.51	36	0.17	35
VU80	2-4-8-16	**0.53**	31	**0.18**	37.24
VU100	2-4-8-16-32	0.51	**24**	**0.18**	**38.92**

16 and a batch size of 4 for 100k iterations. CU20, CU40, CU60, CU80, CU100 are snapshots taken at iterations 20k, 40k, 60k, 80k, 100k, respectively. Training VU is performed with the regularly increasing number of unrolls strategy which is given on Table 5.2. VU20, VU40, VU60, VU80, VU100 are snapshots taken at iterations 20k, 40k, 60k, 80k, 100k, respectively.

Quantitative Results for models CU20-100 and VU20-100 are summarized on Tables 5.7, 5.8.

According to Table 5.7, Accuracy-Robustness performance of the algorithm rapidly increases in the early iterations and reaches to a saturation in the last iterations. VU100, which is the finetuned version of VU80 with 32 unrolls is more robust when compared to VU80 but less accurate. According to EAO metric, performance of these two are equivalent. Expected overlap curves of models VU20-100 are given on Figure 5.11. According to these curves, improvements are observed at almost all sequence lengths as training continues.

According to Table 5.8, Accuracy-Robustness performance of the algorithm did not undergo significant changes after 20k iterations. The best results are obtained at 100k iterations. For both VU and CU, 100k iterations are observed to be sufficient for convergence.

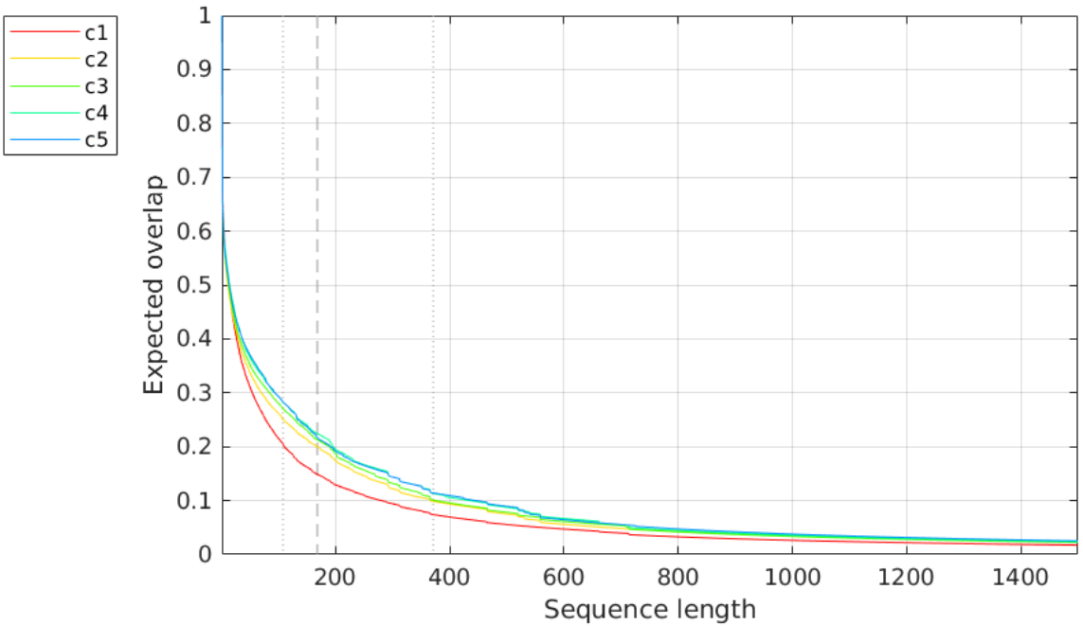

Figure 5.11: Expected overlap curves for models VU20-80 (Best viewed in color).

Table 5.8: Results for Experiment 5, Model CU

Model Name	Iterations	Accuracy	Failures	EAO	Speed (FPS)
CU20	20k	0.51	28	0.16	**39.18**
CU40	40k	**0.52**	25	0.17	38.84
CU60	60k	**0.52**	29	0.16	38.44
CU80	80k	**0.52**	25	0.16	38.73
CU100	100k	0.51	**24**	**0.18**	38.42

5.3.3.6 Experiment 6: Size of Training Data Set

Performance of neural networks is known to be largely dependent on the amount of training data. As a fully supervised algorithm, Re^3 requires training data with bounding box annotations. Bounding box annotations are produced by human experts and it is expensive. Therefore, it is important to know how much the accuracy of Re^3 will be affected when the amount of training data is reduced. With this motivation, we train three models with varying amounts of training data. Each model is trained for 60k iterations with a constant batch size of 4 and number of unrolls equal to 16.

Model DT3 is trained with the entire VID training set containing 3862 videos. Model DT1 and Model DT2 are trained with 400 and 2000 videos which are randomly picked from the complete training set. Training data for these models are described on Table

Table 5.9: Experiment 6: Training Sets

Model Name	Num Videos	Num Tracks	Num Frames	Num Classes
DT3	3,862	7,911	1,122,397	30
DT2	2,000	4,269	946,569	30
DT1	400	787	171,149	30

Table 5.10: Results for Experiment 6: Amount of Training Data

Model Name	Num Videos	Accuracy	Failures	EAO	Speed (FPS)
DT3	3862	**0.52**	29	0.16	38.44
DT2	2000	**0.52**	32	0.15	39.01
DT1	400	0.05	367	0.01	41.65

5.9.

Results for models DT3, DT2 and DT1 are presented on Table 5.10. From Table 5.10 it is observed that performance is improved when the training data is bigger. However, the relationship is not linear. Surprisingly, from 787 tracks model learns almost nothing.

5.3.3.7 Experiment 7: Different RNN types

In this experiment, we compare the performance of Re^3 using different RNN types. Specifically, we train three models with single stage LSTM, GRU and Convolutional LSTM cells. Only Block C is different for these three models. Each model contains 2-stage RNNs with short connection as depicted on Fig. 5.10. Block A is the default Caffenet feature extraction backbone depicted on Fig. 5.7. Block B is a fully-connected layer with 1024 units. Models are trained by applying the training strategy in Table 5.2 up to 80k iterations. Quantitative Results for models G, CL and L are presented on Table 5.11.

According to results in Table 5.11, LSTM performs better than GRU and Convolutional LSTM in terms of accuracy and EAO. On the other hand, GRU and Convolutional LSTM make less failures which is rather interesting. Fastest model is the one with GRU. Convolutional LSTM is significantly slower than the other two models. On the other hand, Convolutional LSTM has the advantage of using small disk space.

Table 5.11: Results for Experiment 7: Different RNN types

Model	RNN Type	Accuracy	Fails	EAO	FPS	Model Size
G	GRU	0.38	**27**	0.15	**39.70**	267.7 MB
CL	Conv LSTM	0.46	**27**	0.16	26.19	**241.0 MB**
L	LSTM	**0.53**	31	**0.18**	37.24	286.6 MB

Table 5.12: Results for Experiment 8: RNN State Reset

Reset Length	Accuracy	Failures	EAO	Speed (FPS)
2	**0.53**	26	0.17	33.59
4	0.50	25	0.17	37.09
8	**0.53**	29	0.17	38.26
16	0.51	**24**	**0.18**	38.42
32	0.52	28	0.17	39.65
64	0.52	112	0.10	**40.34**

To sum up, each RNN type has its own advantage.

5.3.3.8 Experiment 8: RNN State Reset

In [41], when testing a model trained with a maximum unroll of 32, RNN states are reset every 32 frames. By doing so, possible negative effect of truncation in backpropagation is aimed to be eliminated. In this experiment, we test the same model with different LSTM reset lengths. Our aim is to understand the relationship between reset length and system performance. Model CU100 is trained with a constant unroll of 16 and it is tested with various reset lengths. Results are presented on Table 5.12.

From Table 5.12, it is observed that AR performance of the algorithm is optimal when the LSTM reset length is selected to be equal to the number of unrolls applied during training. On the other hand, accuracy is similar for different reset lengths, number of failures significantly increases, when the difference between number of training unrolls and reset length gets larger. A reset length of 32, which is twice the number of unrolls applied during training, does not make a significant difference when compared to 16. On the other hand, when the reset length is 64, the system makes many failures; hence, the performance gets much worse. Based on this observation, it can be concluded that LSTM is able to generalize what it has learned in a limited number

of unrolls (16) to higher number of unrolls (32) up to some degree. LSTM might not be able to generalize to higher reset lengths (64).

Furthermore, it is observed that state resets have a significant effect on the speed of the algorithm. Speed of the algorithm consistently increases when the reset length is increased, which is quite reasonable.

5.4 Experiments with Detection-aided Tracking

For both detection and failure check, Model C_1 presented in Chapter 3 is employed. For tracking, Model CU100 which was presented on Section 5.3.3 is employed. We compare the performances of two detection-aided tracking models constructed with detectors B_1 and C_1 and tracker (CU100) as depicted in Figure 5.5 with a tracker-only baseline model (CU100). It should be noted that while the tracker-only model is a generic tracker, detection-aided model is restricted with classes known by the detector. Therefore, comparison is performed on videos that involve object categories learned by the detector. Furthermore, it should be emphasized that detection-aided model exploits an extra information, namely the category of the object, which is not used in the tracker-only approach.

In detection-aided trackers (B_1+Re^3 and C_1+Re^3), failure check is performed in 5 frame periods. Failure threshold is set to 3e-4, that is, whenever the true class score is below 3e-4, detection is performed. Confidence threshold is set as 0.2 in detection step.

One aim of the experimental evaluation is to investigate whether tracking performance can be improved by incorporating a detection step. Another aim is to compare the context detector and baseline detector in this task as well. Evaluation is performed over 24 VOT 2016 sequences that involve objects from 4 object categories learned by the detector: Person, Car, Cat and Bird. Evaluation is solely based on accuracy in an unsupervised setting which means failed tracks are not reinitialized. Although this setting is not the official evaluation methodology of the VOT benchmark, it is found to be sufficient for demonstrating that improved results are obtained. At each frame, IoU between the ground truth and the track is computed to obtain the accuracy. Accuracies

Table 5.13: Results Obtained with Detection-aided Trackers and Baseline Tracker

Sequence	**Category**	(C_1+Re^3)	(B_1+Re^3)	Re^3	**Est. Fails** (C_1-B_1)
graduate	Person	0.541	**0.592**	0.591	21-10
gymnastics1	Person	**0.200**	0.180	0.118	7-1
gymnastics2	Person	**0.610**	0.571	0.571	4-1
gymnastics3	Person	0.108	0.197	**0.342**	9-4
gymnastics4	Person	**0.555**	**0.555**	**0.555**	0-0
girl	Person	**0.205**	0.150	0.064	3-5
crossing	Person	**0.747**	**0.747**	**0.747**	0-0
bolt1	Person	**0.135**	**0.135**	**0.135**	1-1
bolt2	Person	**0.552**	0.457	0.468	1-1
basketball	Person	0.123	0.062	**0.441**	8-2
soldier	Person	**0.164**	0.137	0.128	21-19
iceskater1	Person	**0.620**	0.514	0.114	4-4
iceskater2	Person	**0.467**	0.183	0.420	3-4
pedestrian1	Person	**0.590**	0.576	0.437	1-1
singer2	Person	**0.649**	0.517	0.390	1-1
wiper	Car	0.103	**0.235**	0.105	29-4
tunnel	Car	0.727	**0.760**	0.727	40-24
car1	Car	0.148	**0.221**	0.092	26-17
car2	Car	0.090	0.06	**0.171**	29-6
racing	Car	**0.719**	0.709	0.710	1-1
birds1	Bird	**0.292**	0.21	0.216	15-16
birds2	Bird	**0.576**	0.569	0.538	1-1
nature	Bird	0.475	0.610	**0.621**	1-0
fernando	Cat	0.444	**0.460**	**0.460**	3-1
All	All	**0.379**	0.358	0.345	231-124

obtained with det+track and track-only scenarios on all sequences are given on Table 5.13. Moreover, number of failures estimated by the check failure blocks are given in the last column.

From Table 5.13, it is observed that detection-aided tracking method has outperformed the tracker-only baseline with both the context detector (C_1) and the baseline detector (B_1). Best results are obtained with the context detector. Also, from Table 5.13 it is evident that failure check with C_1 estimates almost twice as many failures as B_1.

In order to be able to diagnose the weaknesses of our algorithm, it is wise to focus on

the videos on which it performed worse than the baseline model. Visual inspection reveals that for the detection-aided tracker, the main source of failure are the misclassifications obtained in the check failure block. Effects such as motion blurring and occlusion cause disruptions in object appearance which makes it harder to be correctly classified and detected. Another source of failure are the same class objects in the immediate vicinity of the object of interest.

CHAPTER 6

CONCLUSION

6.1 Summary

In this section, we shall summarize the performed work and findings in this thesis. We focused on two well-studied problems of machine vision, which are visual object detection and tracking. On the detection side, we proposed an extension to proposal-based object detection problem which uses local convolutional context features to improve detection performance. On the tracking side, we focused on a recent visual tracking algorithm [41] which combines convolutional and recurrent neural networks. We conducted several experiments to gain insight on the operation of this algorithm. Finally, we proposed a detection-aided tracking model that combines our context based detector with the tracking algorithm. This detection-aided tracker performed better when compared to a tracker-only baseline model.

6.2 Conclusion

Regarding the detection part, conclusions of this thesis can be stated as follows: Experimental results suggest that object detection performance can be improved using local convolutional context features. A separate feature extractor for the context region around the region of interest is essential for this improvement. It is observed that the number of context feature layers and offset ratio significantly affect the performance of a context model. Analysis of false positives reveals that while usage of context features mainly improves localization, it sometimes acts as a source of confu-

sion between different object categories having similar context. Moreover, analysis of sensitivity to object characteristics reveals that context extension yields better results independent of several object characteristics.

On the tracking side, it should be first noted that the overall performance of RNN-based visual trackers is inferior to mature tracking algorithms in the literature. However, RNN-based techniques are still under development with rapid evolution. Several benchmark experiments are conducted to determine how much the accuracy-robustness performance as well as speed and memory allocation of the Re^3 algorithm varies under different configurations. Based on these experiments, following conclusions are reached: Number of unrolls during training, number of feature stages, amount of training data and RNN type being used are factors that have a significant effect on the performance of the Re^3 algorithm. On the other hand; factors such as state reset period applied at test time, the way RNN stages are constructed from basic RNN building blocks, size of the feature vector that summarizes the CNN features have a less significant effect on the performance.

Finally, the proposed detection-aided tracking framework that combines context-based detection model with the RNN-based tracker has a superior performance when compared to both the context-free detection-aided model and the detector-free baseline tracker model.

6.3 Future Work

In this thesis, it was shown that proposal-based detection can be improved by using local convolutional context features. Here we defined the context region by introducing a hyperparameter called offset ratio. The question arises, whether this hyperparameter can be eliminated by e.g. forcing the system to learn to predict the appropriate offset for different objects. Furthermore, the idea of using a seperate feature extractor for the context region can be taken one step further perhaps by using seperate extractors for lower, upper, left and right context regions. Also, experiments with other proposal-based detection methods and other detection benchmark datasets such as MS COCO [110] would be informative.

Tracking performance of the RNN-based baseline model was improved by adding a detection supervision step. However, proposed tracking failure check method has obvious weaknesses against motion blurring and occlusion. One might seek ways to formulate a tracking failure detection method robust to these kind of challenges.

REFERENCES

[1] A. Ouaknine, "Review of deep learning algorithms for object detection," Feb 2018. [Online]. Available: https://medium.com/comet-app/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

[2] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel, "A survey of appearance models in visual object tracking," *ACM transactions on Intelligent Systems and Technology (TIST)*, vol. 4, no. 4, p. 58, 2013.

[3] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[4] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 328–335.

[5] J. Carreira and C. Sminchisescu, "Cpmc: Automatic object segmentation using constrained parametric min-cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1312–1328, 2012.

[6] P. Krähenbühl and V. Koltun, "Geodesic object proposals," in *European Conference on Computer Vision*. Springer, 2014, pp. 725–739.

[7] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr, "Bing: Binarized normed gradients for objectness estimation at 300fps," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 3286–3293.

[8] A. Ghodrati, A. Diba, M. Pedersoli, T. Tuytelaars, and L. Van Gool, "Deepproposal: Hunting objects by cascading deep convolutional layers," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2578–2586.

[9] P. O. Pinheiro, R. Collobert, and P. Dollár, "Learning to segment object candidates," in *Advances in Neural Information Processing Systems*, 2015, pp. 1990–1998.

[10] Z. Jie, W. F. Lu, S. Sakhavi, Y. Wei, E. H. F. Tay, and S. Yan, "Object proposal generation with fully convolutional networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 1, pp. 62–75, 2018.

[11] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[12] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 761–769.

[13] Y. Li, K. He, J. Sun *et al.*, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 379–387.

[14] Y.-H. Tsai, O. C. Hamsici, and M.-H. Yang, "Adaptive region pooling for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 731–739.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[16] S. Gidaris and N. Komodakis, "Object detection via a multi-region and semantic segmentation-aware cnn model," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1134–1142.

[17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[18] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[20] Z. Chen, Z. Hong, and D. Tao, "An experimental survey on correlation filter-based tracking," *arXiv preprint arXiv:1509.05520*, 2015.

[21] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *European conference on computer vision*. Springer, 2016, pp. 850–865.

[22] D. Zhang, H. Maei, X. Wang, and Y.-F. Wang, "Deep reinforcement learning for visual object tracking in videos," *arXiv preprint arXiv:1701.08936*, 2017.

[23] S. Ebrahimi Kahou, V. Michalski, R. Memisevic, C. Pal, and P. Vincent, "Ratm: Recurrent attentive tracking model," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 10–19.

[24] A. Kosiorek, A. Bewley, and I. Posner, "Hierarchical attentive recurrent tracking," in *Advances in Neural Information Processing Systems*, 2017, pp. 3056–3064.

[25] T. Yang and A. B. Chan, "Recurrent filter learning for visual tracking," *arXiv preprint arXiv:1708.03874*, 2017.

[26] "Understanding lstm networks." [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[27] C. Cakebread, "People will take 1.2 trillion digital photos this year - thanks to smartphones," Aug 2017. [Online]. Available: http://www.businessinsider.com/12-trillion-photos-to-be-taken-in-2017-thanks-to-smartphones-chart-2017-8

[28] J. Edwards, "Planet selfie: We're now posting a staggering 1.8 billion photos every day," May 2014. [Online]. Available: http://www.businessinsider.com/were-now-posting-a-staggering-18-billion-photos-to-social-media-every-day

[29] F. Rémy, N. Vayssière, D. Pins, M. Boucart, and M. Fabre-Thorpe, "Incongruent object/context relationships in visual scenes: Where are they processed in the brain?" *Brain and cognition*, vol. 84, no. 1, pp. 34–43, 2014.

[30] t. E. Palmer, "The effects of contextual scenes on the identification of objects," *Memory & Cognition*, vol. 3, pp. 519–526, 1975.

[31] J. L. Davenport and M. C. Potter, "Scene consistency in object and background perception," *Psychological Science*, vol. 15, no. 8, pp. 559–564, 2004.

[32] M. Bar, "Visual objects in context," *Nature Reviews Neuroscience*, vol. 5, no. 8, p. 617, 2004.

[33] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2874–2883.

[34] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *European Conference on Computer Vision*. Springer, 2016, pp. 354–370.

[35] Y. Zhu, R. Urtasun, R. Salakhutdinov, and S. Fidler, "segdeepm: Exploiting segmentation and context in deep neural networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4703–4711.

[36] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition, 2013, pp. 3626–3633.

[37] Y. Ding and J. Xiao, "Contextual boost for pedestrian detection," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2895–2902.

[38] G. Chen, Y. Ding, J. Xiao, and T. X. Han, "Detection evolution with multi-order contextual co-occurrence," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1798–1805.

[39] A. Ali, A. Jalil, J. Niu, X. Zhao, S. Rathore, J. Ahmed, and M. A. Iftikhar, "Visual object tracking—classical and contemporary approaches," *Frontiers of Computer Science*, vol. 10, no. 1, pp. 167–188, 2016.

[40] 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*, vol. 115, no. 3, pp. 211–252, Dec 2015. [Online]. Available: http://dx.doi.org/10.1007/s11263-015-0816-y

[41] D. Gordon, A. Farhadi, and D. Fox, "Re3: Re al-time recurrent regression networks for visual tracking of generic objects," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 788–795, 2018.

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

[43] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[44] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[45] S. Manen, M. Guillaumin, and L. Van Gool, "Prime object proposals with randomized prim's algorithm," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2536–2543.

[46] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International journal of computer vision*, vol. 59, no. 2, pp. 167–181, 2004.

[47] D. M. Greig, B. T. Porteous, and A. H. Seheult, "Exact maximum a posteriori estimation for binary images," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 271–279, 1989.

[48] B. Alexe, T. Deselaers, and V. Ferrari, "What is an object?" in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 73–80.

[49] W. Kuo, B. Hariharan, and J. Malik, "Deepbox: Learning objectness with convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2479–2487.

[50] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges." in *ECCV (5)*, 2014, pp. 391–405.

[51] X. Hou and L. Zhang, "Saliency detection: A spectral residual approach," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007, pp. 1–8.

[52] C. Lu, S. Liu, J. Jia, and C.-K. Tang, "Contour box: rejecting object proposals without explicit closed contours," in *Computer Vision (ICCV), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2021–2029.

[53] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár, "Learning to refine object segments," in *European Conference on Computer Vision*. Springer, 2016, pp. 75–91.

[54] J. Dai, K. He, Y. Li, S. Ren, and J. Sun, "Instance-sensitive fully convolutional networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 534–549.

[55] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2147–2154.

[56] C. Szegedy, S. E. Reed, D. Erhan, and D. Anguelov, "Scalable, high-quality object detection," *CoRR*, vol. abs/1412.1441, 2014. [Online]. Available: http://arxiv.org/abs/1412.1441

[57] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[58] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[59] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy *et al.*, "Deepid-net: Deformable deep convolutional neural networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2403–2412.

[60] T. Kong, A. Yao, Y. Chen, and F. Sun, "Hypernet: Towards accurate region proposal generation and joint object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 845–853.

[61] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, vol. 1, no. 2, 2017, p. 4.

[62] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. [Online]. Available: http://arxiv.org/abs/1612.08242

[63] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[64] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "Dssd: Deconvolutional single shot detector," *arXiv preprint arXiv:1701.06659*, 2017.

[65] Z. Zhang, S. Qiao, C. Xie, W. Shen, B. Wang, and A. L. Yuille, "Single-shot object detection with enriched semantics," Center for Brains, Minds and Machines (CBMM), Tech. Rep., 2018.

[66] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[67] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[68] W. Liu, A. Rabinovich, and A. C. Berg, "Parsenet: Looking wider to see better," *arXiv preprint arXiv:1506.04579*, 2015.

[69] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[70] D. Hoiem, Y. Chodpathumwan, and Q. Dai, "Diagnosing error in object detectors," in *European conference on computer vision*. Springer, 2012, pp. 340–353.

[71] Y. Bar-Shalom and X.-R. Li, *Multitarget-multisensor tracking: principles and techniques*. YBs London, UK:, 1995, vol. 19.

[72] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "Eco: Efficient convolution operators for tracking," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA*, 2017, pp. 21–26.

[73] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE, 2016, pp. 4293–4302.

[74] H. Fan and H. Ling, "Sanet: Structure-aware network for visual tracking," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*. IEEE, 2017, pp. 2217–2224.

[75] B. Han, J. Sim, and H. Adam, "Branchout: Regularization for online ensemble tracking with convolutional neural networks," in *Proceedings of IEEE International Conference on Computer Vision*, 2017, pp. 2217–2224.

[76] H. Nam, M. Baek, and B. Han, "Modeling and propagating cnns in a tree structure for visual tracking," *arXiv preprint arXiv:1608.07242*, 2016.

[77] Z. Teng, J. Xing, Q. Wang, C. Lang, S. Feng, and Y. Jin, "Robust object tracking based on temporal and spatial deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1144–1153.

[78] K. Chen and W. Tao, "Convolutional regression for visual tracking," *IEEE Transactions on Image Processing*, vol. 27, no. 7, pp. 3611–3620, 2018.

[79] T. Zhang, C. Xu, and M.-H. Yang, "Multi-task correlation particle filter for robust object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, no. 2, 2017, p. 3.

[80] Y. Song, C. Ma, L. Gong, J. Zhang, R. W. Lau, and M.-H. Yang, "Crest: Convolutional residual learning for visual tracking," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 2574–2583.

[81] Z. Chi, H. Li, H. Lu, and M.-H. Yang, "Dual deep network for visual tracking," *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 2005–2015, 2017.

[82] H. Fan and H. Ling, "Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking," in *Proc. IEEE Int. Conf. Computer Vision, Venice, Italy*, 2017.

[83] S. Y. J. C. Y. Yoo, K. Yun, and J. Y. Choi, "Action-decision networks for visual tracking with deep reinforcement learning," 2017.

[84] E. Gundogdu and A. A. Alatan, "Good features to correlate for visual tracking," *IEEE Transactions on Image Processing*, vol. 27, no. 5, pp. 2526–2540, 2018.

[85] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, "Learning dynamic siamese network for visual object tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1–9.

[86] R. Tao, E. Gavves, and A. W. Smeulders, "Siamese instance search for tracking," in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE, 2016, pp. 1420–1429.

[87] H. K. Galoogahi, A. Fagg, and S. Lucey, "Learning background-aware correlation filters for visual tracking," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA*, 2017, pp. 21–26.

[88] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *European Conference on Computer Vision*. Springer, 2016, pp. 472–488.

[89] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr, "Staple: Complementary learners for real-time tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1401–1409.

[90] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2544–2550.

[91] G. Zhu, F. Porikli, and H. Li, "Beyond local search: Tracking objects everywhere with instance-specific proposals," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 943–951.

[92] Q. Wang, Z. Teng, J. Xing, J. Gao, W. Hu, and S. Maybank, "Learning attentions: residual attentional siamese network for high performance online visual tracking," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2018.

[93] A. He, C. Luo, X. Tian, and W. Zeng, "A twofold siamese network for real-time object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4834–4843.

[94] J. Choi, J. Kwon, and K. M. Lee, "Real-time visual tracking by deep reinforced decision making," *Computer Vision and Image Understanding*, 2018.

[95] J. S. Supancic III and D. Ramanan, "Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning." in *ICCV*, 2017, pp. 322–331.

[96] C. Huang, S. Lucey, and D. Ramanan, "Learning policies for adaptive tracking with deep feature cascades," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2017, pp. 105–114.

[97] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[98] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015, pp. 2048–2057.

[99] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[100] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[101] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[102] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, vol. 3. IEEE, 2000, pp. 189–194.

[103] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[104] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[105] D. P. Kingma and J. L. Ba, "Adam: Amethod for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

[106] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Cehovin, G. Nebehay, G. Fernandez, T. Vojir, A. Gatt *et al.*, "The visual object tracking vot2013 challenge results," in *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*. IEEE, 2013, pp. 98–111.

[107] L. Čehovin, A. Leonardis, and M. Kristan, "Visual object tracking performance measures revisited," *IEEE Transactions on Image Processing*, vol. 25, no. 3, pp. 1261–1274, 2016.

[108] ——, "The visual object tracking vot2015 challenge results," in *Visual Object Tracking Workshop 2015 at ICCV2015*, Dec 2015.

[109] ——, "The visual object tracking vot2016 challenge results," in *Visual Object Tracking Workshop 2016 at ICCV2016*, Dec 2016.

[110] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.