Object Detection in Maritime Images

Exploring Deformable Convolutions for Object Detection

Master's thesis in Electrical Engineering

Jesper Andersson

Department of Electrical Engineering

Master's thesis 2019

Object Detection in Maritime Images

Exploring Deformable Convolutions for Object Detection

Jesper Andersson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2019

Object Detection in Maritime Images
Exploring Deformable Convolutions for Object Detection
Jesper Andersson

Supervisor: Amir Shahroudy, Electrical Engineering
Advisor: Eren Erdal Aksoy, Volvo Group Trucks Technology
Advisor: Ethan Faghani, Volvo Penta
Examiner: Fredrik Kahl, Electrical Engineering

Master's Thesis 2019
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2019

Object Detection in Maritime Images
Exploring Deformable Convolutions for Object Detection
Jesper Andersson
Department of Electrical Engineering
Chalmers University of Technology

Abstract

In this thesis, we have used the Faster R-CNN object detection framework to localize and classify boats in a small dataset of maritime images. In addition to the maritime images experiments, we have explored the interplay of deformable convolutions with other techniques that affect the spatial sampling locations of convolutional filters.
For the deformable convolution experiments, we examined the results of applying atrous spatial pyramid pooling and an inception-like block of multiple filter sizes to a regular convolutional network as well as a deformable convolutional network.
For the maritime images experiments, we have evaluated the effect on mean average precision, training speed and test speed for the following techniques, deformable layers, data augmentation, online hard example mining, soft-NMS and multi-scale testing. The results showed that adding deformable layers significantly improved the model with an increase in mean average precision of 5.8 at the intersection over union threshold 0.7.

Keywords: Deformable Convolutions, Object Detection, Convolutional Neural Networks, Maritime Images, Faster R-CNN

Acknowledgements

First of all, I would like to thank Volvo Penta for offering me the opportunity to do my master's thesis in this interesting and quickly developing area. A special thanks goes to Volvo Penta Krossholmen for helping me to collect the maritime images.
I would also like to thank my thesis advisors Eren Erdal Aksoy and Ethan Faghani at Volvo. They have always been available and offered guidance and assistance when needed. In addition, I would like to express my gratitude to my academic supervisor Amir Shahroudy at Chalmers for his valuable comments on the thesis and his technical advice. I would also like to thank my examiner Fredrik Kahl.
Finally, I would like to thank my wife for all her support and encouragement.

Jesper Andersson, Gothenburg, June 2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Docking is one of the most stressful experiences boat owners face. To make docking easier and safer, Volvo Penta has created a self-docking solution called "Easy-docking". However, the current system requires sensors on the boat as well as the berth to guide the boat to its docking position. Ideally, the docking system should be able to rely entirely on sensors on the boat such that it can dock at any berth. One step towards the goal of docking safely anywhere is to localize and detect objects in images, i.e. object detection.

In this thesis we examine what accuracy can be achieved when detecting boats in a small dataset of maritime images. Furthermore, we explore the interplay between deformable convolutions and other techniques that affects the spatial sampling points of convolutional filters.

1.1 Thesis Outline

This thesis comprises six chapters. In the first chapter we provide a brief introduction to machine learning, neural networks, object detection and deformable convolutions. Furthermore, we state the goal of the thesis.

The second chapter provides the reader with sufficient background knowledge in the field of neural networks, object detection and deformable convolutional networks to understand the techniques and models that have been used in this thesis. The chapter ends with describing the datasets that have been used.

The background chapter is followed by a chapter describing the methods that have been used. First, we list the hardware and describe the implementation framework that has been used for the experiments. Secondly, we describe the experiments that have been performed in terms of model construction, model tuning and training protocol. Thirdly, we explain the mean average precision metric that is used to evaluate object detection models.

The fourth chapter presents the results of the experiments described in the methods chapter. Following the results chapter, we analyze the results and suggest ideas for further work in a discussion chapter. We conclude the thesis with a conclusion chapter where we give a brief summary of the findings in this thesis.

1.2 Machine Learning and Neural Networks

Since decades ago, computers have put humans to shame in terms of raw computational ability. However, computers have only recently started to match human performance in tasks that involve dealing with high-dimensional data such as images. While a human can recognize an object in an image with a single glance, it is hard to describe the task of object detection to a computer in a formal way. By using machine learning techniques, we do not need to provide any such formal rules. Instead, we feed the computer with examples from which it can extract patterns.
Traditional machine learning techniques combined with hand-engineered features have seen some success in the computer vision area[1]. Nevertheless, it was not until the advent of neural networks that the computer systems started to approach human level performance for computer vision tasks such as image recognition[2]. One of the reasons that neural networks works so well is that their performance continually improves with the number of training examples, whereas the performance of many traditional machine learning methods seems to saturate after a certain point. Another reason that neural networks work so well is that they have a layered structure that allows the network to learn increasingly more complex concepts. This layered structure of more and more complex representations allows neural networks to learn useful features from raw input data, such as the pixels of an image, as opposed to other traditional machine learning methods which first need to extract hand-engineered features from the input. The end to end learning from raw pixels to the desired output, not only removes the time-consuming step of finding good hand-engineered features, but also allows the neural network to learn even better features. This nested hierarchy of more and more complex representations is a special type of machine learning referred to as deep learning.

1.3 Object Detection

Object detection is the task of classifying all objects of interest in an image and drawing bounding boxes around them.
This task has traditionally been done by sliding a window over the image and extracting features such as histogram of oriented gradients [3] or Haar features[1] which are then passed to a classifier. However, the field has completely changed since convolutional neural networks[4] were popularized for computer vision tasks by the groundbreaking work of Krizhevsky et. al.[5]. The R-CNN object detection system[6] used Krizhevsky's convolutional neural network to extract features from region proposals and improved the mean average precision on the PASCAL VOC[7] benchmark dataset by more than 30%. Since then, extensions of the R-CNN named Fast R-CNN[8] and Faster R-CNN[9] have improved the accuracy and inference time to a great extent.
It it is hard to compare the accuracy of different object detection systems since there is a large number of general techniques that can be used to improve all detectors. These techniques include but are not limited to multi-scale testing, online hard example mining[10], heavier heads, deformable convolutions[11], test time augmentation, deeper backbone architectures and feature pyramid networks[12]. In

recent years, the most commonly used benchmark dataset for object detection is the COCO[13] dataset. TridentNet[14], SNIPER[15] and PANet[16] are three of the top-performing detectors on this dataset. All of these detectors use deformable convolutions and a region-based "Faster R-CNN"-like detection framework.
In addition to the aforementioned region-based detectors, there is another type of object detection system called single shot detector. Two of the most notable single shot detectors are Single Shot MultiBox Detector (SSD)[17] and You Only Look Once (YOLO)[18]. Compared to region-based detectors, single shot detectors are generally faster but lack in accuracy.

1.4 Deformable Convolutions

Convolutional neural networks draws inspiration from the animal visual cortex and makes use of a layered structure of learnable filters that aggregates the value of spatially close values. The aggregation is done by doing a convolution between the image and the weights in the convolutional filter, hence the name convolutional neural networks. In the case of classifying objects in an image, we can conceptually think of the filters in the first layer of a convolutional neural network as being able to detect edges from nearby pixel values. The filters in later layers can then detect curves from nearby edges. From spatially close curves the filters of even deeper layers can then detect object parts. Finally, the object parts can be used to determine object identity.
In standard convolutional filters, the spatial sampling positions that we aggregate over are fixed and are typically a 3×3 area of adjacent positions. Depending on the scale, shape and other geometric properties of the object that we are trying to detect, this configuration may not be optimal. For large objects we would like the sampling locations to be located further away from each other and for tall and narrow objects we would like the spatial sampling locations to follow the shape of the object. Deformable convolutional networks[11] address the question of finding good spatial sampling locations by presenting a deformable convolution module with learnable offset parameters for the spatial sampling positions. The offset parameters are conditioned on the values of the feature map where the convolution is taking place so that it can adapt to the shape of the object.

1.5 Problem Statement

The main goal of this project is to detect boats in a small dataset of images in a maritime environment. We will examine the performance when using the Faster R-CNN object detection framework and employing a range of techniques, including deformable convolutions, data augmentation, online hard example mining, soft-NMS and multi-scale testing.
The second goal of this project is to look deeper into deformable convolutional networks and explore the interplay of deformable convolutions with other techniques that affects the spatial sampling locations of the convolutional filters.

Chapter 2

Background

In this section we present the necessary background to understand the methods this thesis builds upon. In the first sections we discuss feedforward networks, convolutional neural networks and methods to train these networks. In the later sections we cover the topic of object detection and some of the most common object detection systems. We continue by describing deformable convolutional networks. Finally, we present the datasets that have been used in this thesis.

2.1 Neural Networks

Neural networks[19] are used as non-linear function approximators. In the case of object detection, the function that we are trying to approximate is a mapping from the pixel values in an image to the class and bounding box coordinates of all objects of interest in that image. The networks draw inspiration from the structure of the human brain and consists of a large number of connected artificial neurons.

2.1.1 Artificial neurons

Artificial neurons, also termed units, are information-processing units which can be said to be made up of three basic elements, see Figure 2.1.

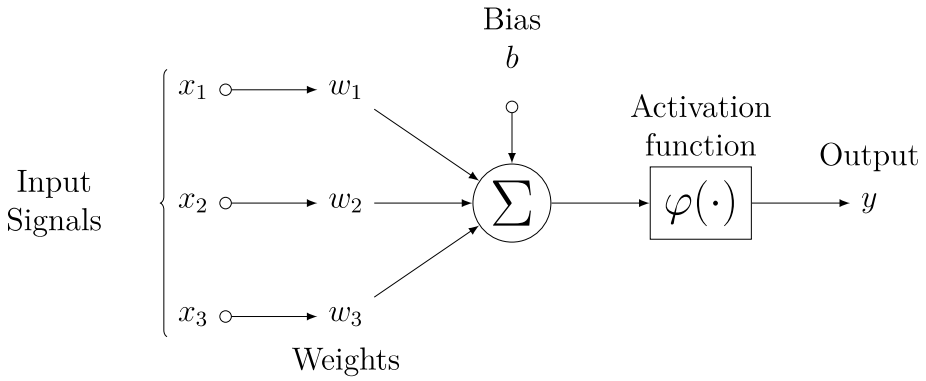


Figure 2.1: Model of an artificial neuron

The first element is the input signals x. Each input signal x_i is multiplied by a weight w_i and given to the second element of the neuron, the adder. The adder adds up all the weighted input signals together with a bias b which has the function of increasing or decreasing the net input to the neuron. The net input z can be described mathematically as:

$$z = b + \sum_i x_i \cdot w_i \tag{2.1}$$

Finally, the net input is given to the third element, an activation function, which computes the output of the neuron. The output y can then be written as:

$$y = \varphi(z) \tag{2.2}$$

Where $\varphi(\cdot)$ is the activation function. The activation function for neurons that are used to produce the final output of a neural network is task-dependent. For regression tasks, such as the coordinates and size of an object in an image, the identity function is commonly used. For multi-class classification of N-classes, the softmax activation function is used over N neurons, where each neuron outputs the probability for a specific class:

$$\varphi_i(\overrightarrow{z}) = \frac{e^{z_i}}{\sum_{k=1}^{N} e^{z_k}} \quad \text{for} \quad i = 1, \ldots, N \tag{2.3}$$

The activation function for units that calculates intermediate results, i.e. *hidden* units, is typically a non-linear function such as the rectifier:

$$\varphi(z) = \max(0, z) \tag{2.4}$$

Units with the rectifier as activation function are commonly known as rectified linear units or simply ReLU[20].

2.1.2 Feedforward Neural Networks

The feedforward neural network is a simple type of neural network that form the basis of more advanced architectures, such as the convolutional neural network and recurrent neural network. Feedforward neural networks consists of several layers of neurons and the name feedforward comes from the information flow going in one direction, where the outputs of one layer are fed as input to the next layer. The first layer is called the input layer and the last layer is the output layer. Between the input layer and the output layer there are zero or more hidden layers which performs intermediate computations. The Figure 2.2 shows a feedforward neural network with two hidden layers.
Although the representational power of a single neuron is limited, the universal approximation theorem states that a feedforward neural network with enough hidden units can approximate any continuous function. In order to utilize this capacity and approximate a target function, the neural network must undergo training.

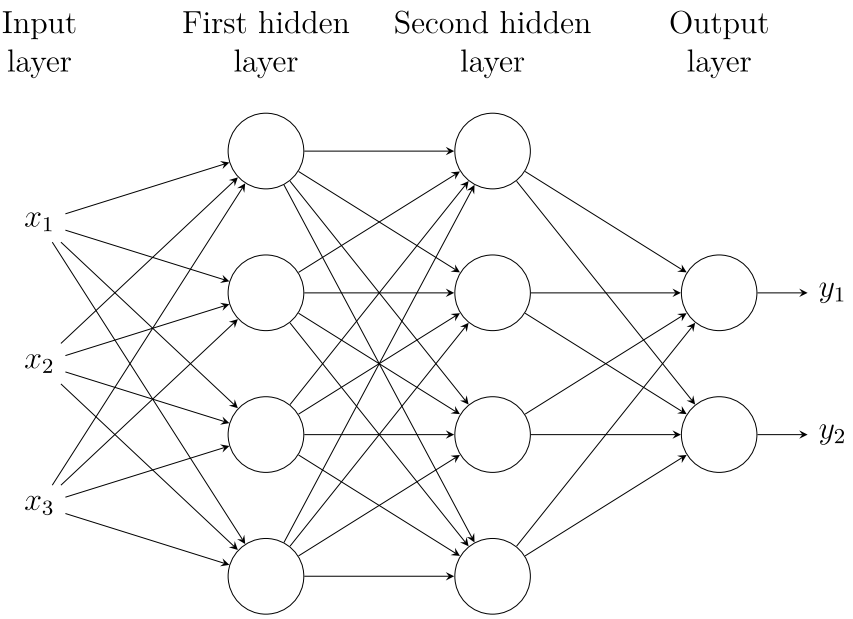


Figure 2.2: Feedforward network with two fully connected hidden layers, three inputs and two outputs. Each circle represents an artificial neuron.

2.1.3 Training

The most common learning paradigm used for training neural networks is supervised learning. In supervised learning we are given a training set containing input examples annotated with ground truth output values. When training a neural network, the weights of the network are tuned to approximate the mapping defined by the training data.

Cost Function

A cost function, also referred to as a loss function, quantifies the performance of a network. The cost function varies from task to task. For bounding box regression in object detection the cost function can be defined as a sum of the smooth L_1 loss for the width, height and center coordinates of the bounding box. The smooth L_1 loss is defined as:

$$\text{smooth}_{L_1}(y, \hat{y}) = \begin{cases} 0.5(y - \hat{y})^2 & \text{if } |y - \hat{y}| < 1 \\ |y - \hat{y}| - 0.5 & \text{otherwise} \end{cases} \tag{2.5}$$

Where $\hat{y}$ is the value predicted by the model and y is the ground truth output value of the training data.
For multi-class classification of N-classes the loss L for each training example is given by the negative log probability for the ground truth class i:

$$L(\hat{y}) = -\log \hat{y}_i \tag{2.6}$$

Given a cost function, we can use the backpropagation algorithm[21] to find the cost derivatives w.r.t. the weights in the network.

Backpropagation

The backpropagation algorithm[21] is an algorithm for computing gradients in neural networks and can be said to consist of two phases, the forward propagation phase and the backward propagation phase. In the forward propagation phase, the intermediate activation values and the output is computed by propagating the input through the network. In the backward propagation phase, the chain rule for derivation is used iteratively in combination with the cost function and the previously computed activation values to give us the cost derivative w.r.t. all the weights in the network.
When the cost derivatives are computed, we can then use a gradient based learning method to update the weights and minimize the cost function, i.e. train the network.

Gradient based optimization

The simplest type of gradient based optimization methods is gradient descent. Gradient descent, also known as "batch" gradient descent, is a simple optimization method where we iteratively update the parameters of our network θ by a small step in the direction of the negative gradient of the cost function C in θ_n, where θ_n represents the parameters at iteration n. These updates are repeated until θ_n has converged to a local minimum:

$$\theta_{n+1} = \theta_n - \alpha \nabla C(\theta_n) \tag{2.7}$$

Where α is the learning rate and determines the step size. The learning rate is one of the most influential hyperparameters in a neural network. If the learning rate is small, convergence may take a long time. On the other hand, if the learning rate is too large, the objective function may not converge at all.
Gradient descent calculates the cost function based on the entire training set in each iteration and is generally too slow for practical use in neural networks. Stochastic gradient descent (SGD) works in a similar fashion as gradient descent. However, when using SGD we do not consider the entire set of training examples for each iteration but rather a small randomly chosen subset. This small subset of the training examples is known as a "mini-batch" and may be as small as a single training example. Gradient descent form the basis of more advanced gradient based optimization methods such as Adam[22].

Regularization

Training a neural network minimizes the cost function with respect to a training set. However, our end goal is to perform well during inference time on unseen examples. Techniques that are employed to allow a model to generalize better rather than reducing the training error are referred to as regularization techniques. One common regularization technique is to use weight decay. Weight decay multiplies the weights of the network by a factor of slightly less than 1 in each iteration of the gradient based learning algorithm, penalizing large weights. Another technique to make the model generalize better is to augment the training data with synthetic examples. In the case of image classification, transformations of the input such as

flipping, rotation, scaling and translation can often be used without changing the correct class.

2.1.4 Convolutional Neural Networks

Convolutional Neural Networks[4] (CNNs) are a class of neural networks that are used for grid-like data where there are strong local connections between the inputs. Convolutional networks are most commonly seen in computer vision applications, but they have also been succesfully applied to time-series data[23].

Convolutional Layer

A convolutional layer consists of several learnable filters which are convolved across the input data. The density of the positions of the input data that the filters will be convolved over is determined by the stride parameter. With a stride of one the filters are applied at every position, with a stride of two the filters are applied at every other position and so on. When convolving a filter F of dimension $m \times n$ over a single position of a 2-D input data I at row i and column j, the weights of the filter are multiplied elementwise with the input and then added together to produce a single output value:

$$S(i,j) = \sum_m \sum_n I(i+m, j+n)F(m,n) \tag{2.8}$$

By doing this for all positions we get a 2-D output feature map. See figure 2.3 for an illustration of a single 2×2 filter convolved over a 3×3 2-D input feature map. For each filter, its 2-D output feature map is stacked into a 3-D tensor. In general, we apply 3-D filters over the 3-D stack of feature maps, but the principle is the same as in the 2-D case. A convolutional layer is typically followed by a non-linearity layer which applies a non-linear function such as the rectifier.
One of the main advantages of convolutional layers is that they are very parameter efficient. The parameters for a single filter is shared over all positions that the filter is convolved over. This parameter sharing combined with the fact that the filters are generally much smaller than the input, gives us that the number of parameters in a convolutional layer is several orders of magnitude smaller than in a fully connected layer. This is especially important when working with high dimensional inputs such as images.

Pooling Layer

Pooling layers are used to reduce the dimension of the activation maps of convolutional layers and to make the features more robust to small translations. This is achieved by aggregating clusters of adjacent values of the activation maps. The intuition behind the translation invariance is that the aggregated values will remain similar even if there are small changes to the input. The most commonly used aggregation function is to take the maximum value of the cluster, this is called max-pooling[24]. Figure 2.4 shows the result of applying max-pooling with a filter size of 3×3 and stride 2 to a feature map of size 3×3.

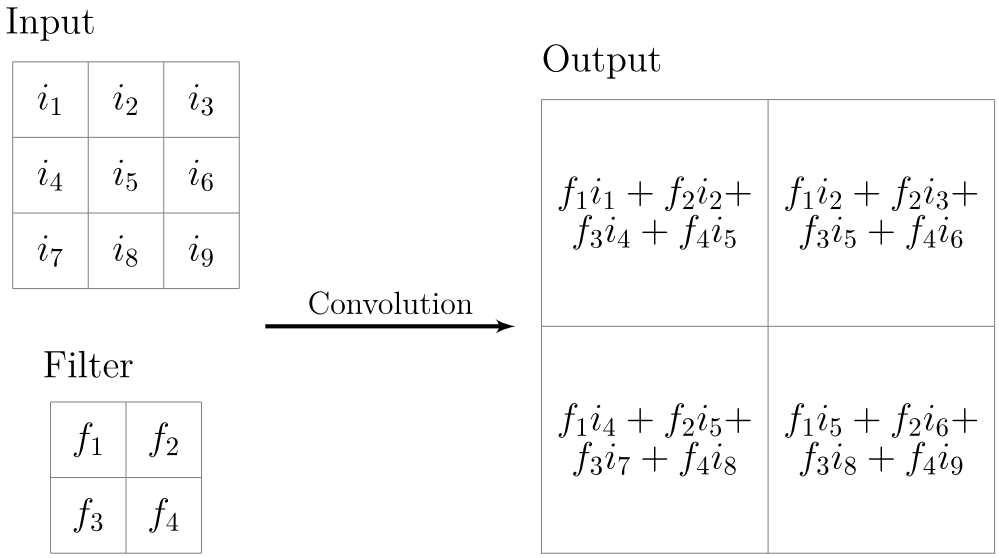


Figure 2.3: Example of 2-D convolution with filter size 2×2 and stride 1 applied to a 3×3 input feature map. The output is a 2×2 feature map.

Convolutional Architectures

One of the first applications of convolutional networks was developed by LeCun et. al.[25] in 1998 to classify hand-written digits. The architecture used was called LeNet and comprises 7 layers. It starts with three convolutional layers interspersed with two pooling layers and is followed by a fully connected layer and a softmax output layer.

Although there has been some successful applications of convolutional networks following the LeNet network, it was not until AlexNet[5] won the ImageNet Large Scale Visual Recognition Challenge[2] (ILSVRC) in 2012 that convolutional networks were widely popularized in computer vision. The structure of AlexNet closely resembles that of LeNet. The key difference is the depth of the network, where AlexNet has more convolutional and fully connected layers. The intuition behind more layers working better is that deeper layers learn increasingly complex representations. In the first layers we can easily identify edges, and as we move up the layers we can recognize corners, contours and more complex object parts. The value of depth has been further demonstrated by the even deeper convolutional network VGGNet[26], which achieved second place in the 2014 ILSVRC. To train the deep and computationally expensive network, AlexNet utilized GPUs to speed up training.

Even though the depth of an architecture is important for performance, it comes with some optimization problems. When using the backpropagation algorithm, the gradient is propagated backwards from the output layer to the earlier layers. If the network is very deep, the gradient signal may become very small for the earlier layers, which is called the vanishing gradient problem. To tackle the vanishing gradient problem, Kaiming He et. al. presented the ResNet[27] architecture. ResNet uses skip connections which allows the gradient to be passed back to previous layers unchanged, maintaining a strong signal even at earlier layers. Using skip connections and stacking these residual block allows for networks with hundreds or even

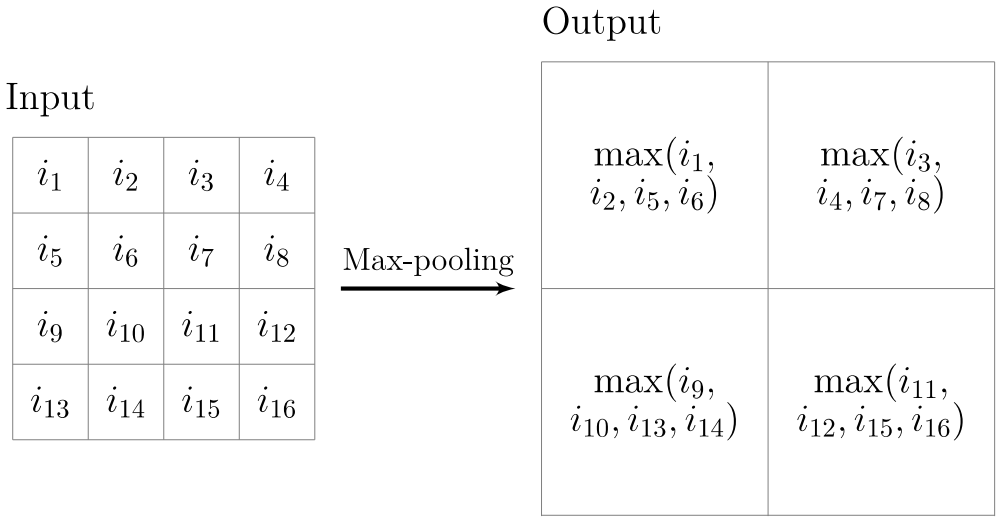


Figure 2.4: Example of max-pooling with filter size 2×2 and stride 2 applied to a 4×4 input feature map. The output is a 2×2 feature map.

thousands of layers. ResNet performs remarkably well and was the winner of the 2015 ILSVRC.
The convolutional network architectures mentioned above are used for image recognition, but they they can also be used in object detection systems as feature extractors.

Transfer Learning in Convolutional Networks

Transfer learning is a technique where we store knowledge that we acquire by learning one task and then apply this knowledge to solve another task. The first stage is often referred to as pre-training and the second stage is referred to as fine-tuning. Transfer learning is not limited to images and convolutional neural networks, but has turned out to be a core component in systems that solve computer vision tasks.
Transfer learning makes sense when we have little data for the problem that we are trying to solve and a large amount of data for the problem that we are trying to transfer knowledge from. Feature extractors are generally trained on the ImageNet dataset which has more than a million images and thousands of classes.
The reason that transfer learning works for convolutional neural networks is that the layers in the pre-trained network learn general features that are good at detecting a wide range of objects. This is especially true for the earlier layers which detects edges, curves and parts of objects. These general features are probably useful for the new task as well. The usefulness of the high-level features in the later layers of a pre-trained network is determined by the similarity of the original and new task.

2.2 Object Detection

Object detection is the task of classifying all objects of interest in an image and drawing bounding boxes around them, see figure 2.5.

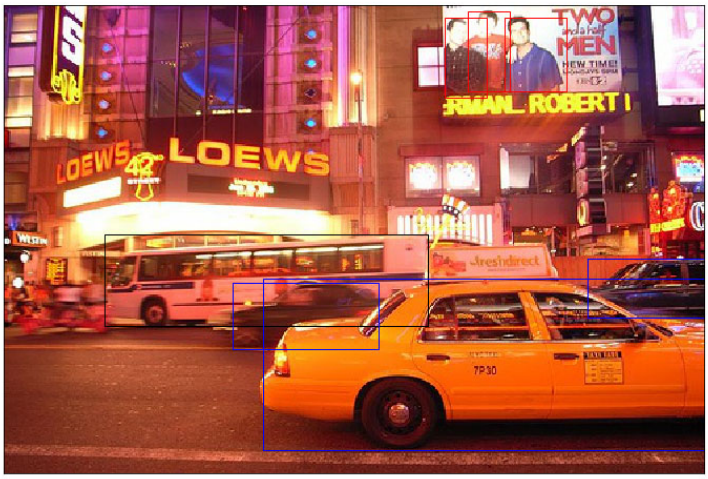


Figure 2.5: Sample image of PASCAL VOC dataset with ground truth bounding boxes around objects of interest.

The modern history of object detection took off in 2013 when convolutional networks started to be used in object detection systems. Most convolutional based object detection systems can be categorized into two-stage detectors and one-stage detectors. In this section we go through the state-of-the art detectors for the two types of object detection systems and give a brief mention to the classic methods preceding them.

2.2.1 Classic Object Detectors

Before the advent of modern object detectors, the sliding window algorithm was commonly used. The sliding window algorithm works by sliding a fixed size rectangular region across an image. For each region, features are extracted and then passed to a classifier. LeNet[25] is one of the early successful object detectors which uses the sliding window algorithm for hand-written digits. The Viola-Jones[27] is another influential sliding window detector and was the first real-time object detection framework for face detection. Viola-Jones uses Haar features and a cascade of gradually more complex classifiers. Another significant step in the history of object detection is the work of Dalal and Triggs[3] which demonstrates the effectiveness of HOG descriptors as features for pedestrian detection.
Deformable part models[28] (DPMs) was the best performing models before the modern convolutional object detectors and held the top results on the canonical PASCAL VOC[7] object detection benchmark dataset for many years. The DPMs model an object as a set of parts. The model expect the parts to have a certain arrangement of relative locations and weighs the likelihood of relevant parts being present against the parts deviation from the expected spatial arrangement, i.e. deformation.

2.2.2 Two-Stage Detectors

Two-stage detectors are the most accurate detectors to date. In the first stage, a set of proposal regions are generated. These proposals are then passed to the second stage for classification and bounding box refinement. Girshick et. al. invented the first two-stage detector, called R-CNN: Regions with CNN features[6].

R-CNN

The R-CNN[6] object detection system set the starting point for a new era in object detection. After years of stagnating results on the PASCAL VOC benchmark dataset, R-CNN improved the mean average precision (mAP) by more than 30%. R-CNN uses the Selective Search[29] region proposal algorithm to generate approximately 2000 proposals. The proposals are then warped into a fixed size and fed to a convolutional network for feature extraction. The original R-CNN paper presents results with AlexNet and VGG-16 as feature extractors. However, any kind of convolutional architecture used for image classification can also be used as a feature extractor for almost any kind of two-stage or one-stage object detector. Classification in the R-CNN is done with category-specific support vector machines (SVMs) and a linear regression model is used for refinement of bounding boxes. Figure 2.6 depicts the architecture of the R-CNN detector.

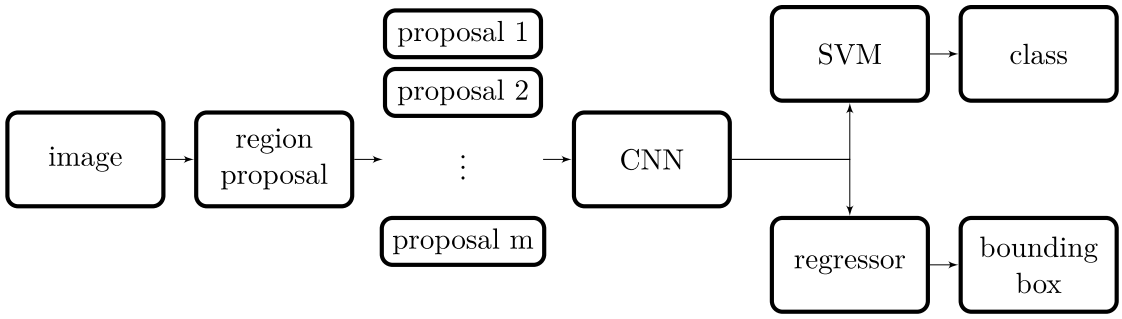


Figure 2.6: Architecture of R-CNN.

Fast R-CNN

Fast R-CNN[8] is an extension of the R-CNN which is much faster and has higher detection quality. Instead of processing each region proposal through a convolutional network, Fast R-CNN processes the entire image through a convolutional network to produce feature maps for the whole image. The region proposals are then used to extract regions from the feature maps instead of the image. These regions of the feature maps are then passed to a pooling layer to create regions of interest (RoIs). The RoIs are then fed to fully connected (FC) layers which branches into a bounding box regression layer and a softmax classification layer. Figure 2.7 depicts the architecture of the Fast R-CNN detector.

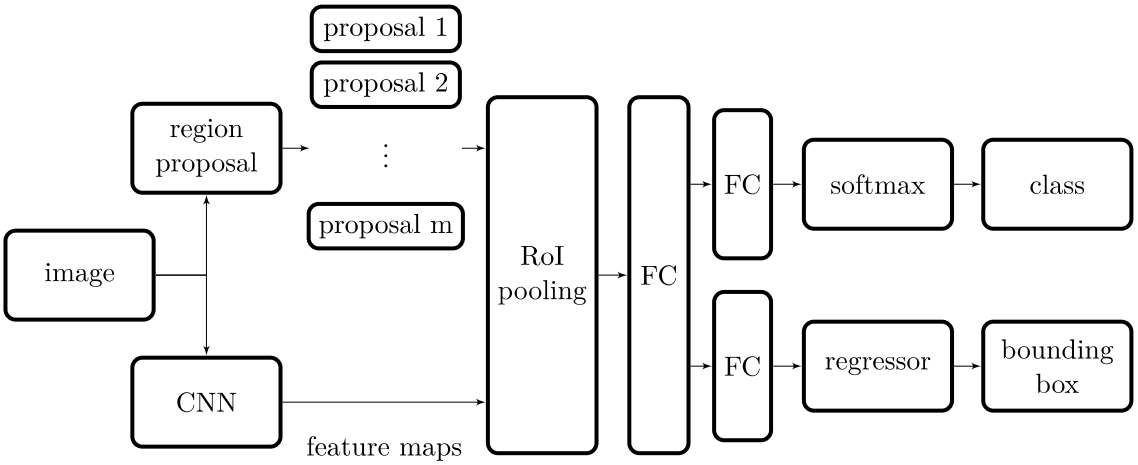


Figure 2.7: Architecture of Fast R-CNN. Compared to R-CNN, the feature extraction is done once for each image instead of each proposal. This gives a significant speedup.

Faster R-CNN

Faster R-CNN[9] is similar in structure to Fast R-CNN but makes use of a convolutional network for region proposals, called region proposal network (RPN). The proposal network share convolutional layers with the feature extraction network and generates proposals by using the sliding window technique over the feature map of the last convolutional layer. The features of each sliding window is passed to two sibling networks to produce regressed region proposals with objectness score for that windows position, where objectness is a measure for the likelihood of the proposal containing an object. The classification and bounding box regression part of the architecture is basically the same as in Fast R-CNN. Faster R-CNN is much faster and has higher detection accuracy than Fast R-CNN. Figure 2.8 depicts the architecture of the Faster R-CNN detector.

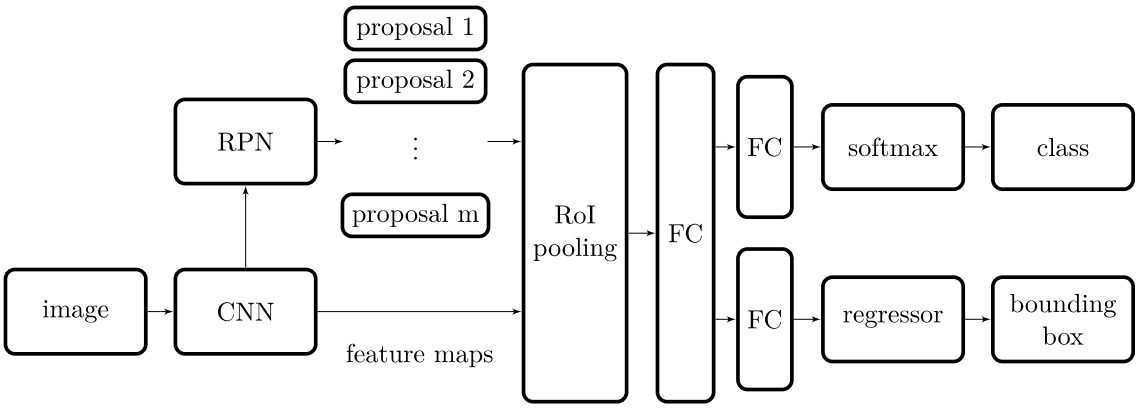


Figure 2.8: Architecture of Faster R-CNN. The region proposal is done by a separate region proposal network which uses the feature maps from the feature extraction process, improving the training and inference time compared to Fast R-CNN.

Faster R-CNN and extensions of Faster R-CNN are the most accurate object detectors. The winning submission[30] of the 2017 COCO[13] detection challenge was based on the Faster R-CNN extension called Mask R-CNN[31]. Mask R-CNN extends Faster R-CNN with an object mask branch to be able to solve another task, namely object instance segmentation. The performance gain of Mask R-CNN over Faster R-CNN comes from trying to solve multiple tasks simultaneously, which can have a regularizing effect on the model and is known as multi-task learning[32]. This thesis will not cover instance segmentation and will not delve deeper into Mask R-CNN.

2.2.3 One-Stage Detectors

One-stage detectors are optimized for speed and are not as accurate as two-stage detectors. They are called one-stage detectors since they do not use a region-proposal step but rather predict the bounding boxes and object classes with a single convolutional neural network directly. Two of the most succesful one-stage detectors are YOLO[18] and SSD[17]. Another influential one-stage detector is RetinaNet[33], which demonstrates the effectiveness of the focal loss function and feature pyramid networks.

YOLO

You Only Look Once (YOLO) [18] is an object detection system that is extremely fast and divides the image into a grid. Each cell in the grid is responsible for predicting bounding boxes for objects where the center of the object is located inside of the grid cell. The original architecture is called DarkNet and comprises 24 convolutional layers and 2 fully connected layers. After the original version of YOLO, two improved versions have been released, namely YOLOv2[34] and YOLOv3[35]. The improvements include incorporating ResNet-like residual blocks and usage of dimension priors for the bounding boxes called anchors.

SSD

As we move down the convolutional layers of a convolutional network, we have that the spatial resolution decreases. This makes it hard to detect small objects in the low resolution layers deep down in the convolutional network. The Single Shot MultiBox Detector (SSD) [17] is a one-stage detector that tries to deal with this problem by predicting objects from feature maps at multiple levels of the network. One of the issues with this method is that the features of the high resolution feature maps are semantically weaker. To enrich the semantically weaker layers we can use top-down connections as is done in the RetinaNet architecture.

RetinaNet

The main contribution of RetinaNet[33] is the use of a novel loss function called focal loss. The loss function reduces the loss of well-classified easy examples to

focus on misclassified hard examples. This focus is especially important for one-stage detectors which has an extreme background-foreground class imbalance with many easily classified background examples.
Other than focal loss, RetinaNet also makes use of a component called feature pyramid networks[12] to detect objects at different scales. Similarly to how SSD works, feature pyramid networks uses feature maps at multiple levels of the network to make predictions. The difference is that it also uses top-down connections to propagate information from the deep and semantically rich feature maps to the shallow but semantically weaker feature maps. This allows for semantically strong feature maps at multiple spatial resolutions. The feature pyramid technique is not limited to one-stage detectors but can be used in the feature extraction step of a two-stage detector as well. Figure 2.9 illustrates how feature pyramid networks works.

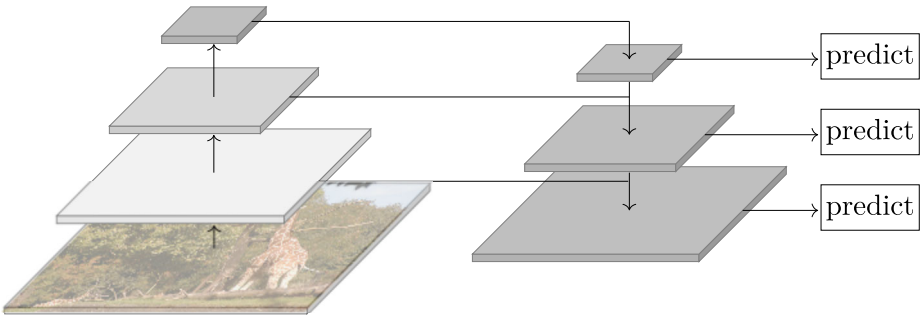


Figure 2.9: Feature Pyramid Network. The bottom-up pass is similar to how a regular feature extractor work. However, the top-down and lateral connections allows semantically strong feature maps at multiple spatial resolutions. The feature maps are colored such that semantically stronger features have a darker color.

2.3 Deformable Convolutional Networks

In this section we cover deformable convolutional networks [11]. The deformable modules in a deformable convolutional network can be used to replace regular convolutional layers and RoI-pooling layers in any object detection system to learn invariance to geometric transformations. At the cost of a few extra parameters for the deformation offsets, deformable convolutional networks produce leading results on the COCO benchmark for object detection[36].

2.3.1 Geometric Transformations

Geometric transformations such as scale, rotation, translation and other kinds of warping is one of the main challenges of object detection. One way to deal with this problem is to collect or create data with enough variation. Another way is to use feature detection algorithms invariant to transformations such as SIFT (scale invariant feature transform) [37]. Deformable part models [28] mentioned in section 2.2.1 tackle this problem by learning spatial deformation through weighing the

deformation between parts against the individual parts score of being part of the object. More recently, spatial transformer network[38] introduces a learnable spatial transformer module that warps the feature map in a global manner.

Deformable convolutional networks use deformable convolutions and deformable RoI-pooling to achieve transformation-invariance. Deformable convolutions share some of the ideas of the spatial transformer module, but it uses local and dense transformations of the feature maps instead of a global transformation. Deformable RoI-pooling is similar in spirit to deformable part models and learns deformation between entire parts.

2.3.2 Deformable Convolutions

Deformable convolutions add learnable offset parameters to the spatial sampling locations of convolutional filters, see figure 2.10.

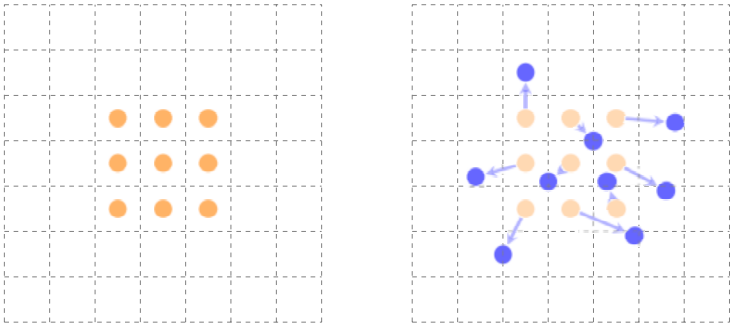


Figure 2.10: Illustration of the spatial sampling locations in a convolutional 3×3 filter. The left image illustrates the sampling locations of a regular convolutional filter and the right image illustrates a deformable filter with offsets represented by the arrows.

The motivation behind modifying the sampling locations is that we can alter the receptive field of a convolutional network. The receptive field is the area of the input that a neuron receives information from. For deep neural networks, the receptive field could in theory be the entire image. However, pixels near the center provide more information and the effective receptive field is much smaller. Deformable convolutions are conditioned on the feature maps of the input to give an effective receptive field matching the object that the neuron is centered at, see figure 2.11.

Deformable convolution can replace a regular convolutional layer by performing the following steps. Firstly, we introduce a parallel convolutional layer which generate offsets for each position of the input feature map that we will convolve over. Secondly, we use these offsets in our existing convolutional layer to modify the sampling locations for each position that we convolve over. Figure 2.12 illustrates the information flow with the introduction of a parallel offset branch.

Figure 2.11: The red dots show the sampling locations of the final three layers of a deformable convolutional network for the neuron located at the green dot. In the left image we can see that the neuron that is representing the location of the green dot has sampling locations that are adjusted to the object at that position, namely a person. Likewise, the model has learned how to adjust the sampling locations for the boat in the right image.

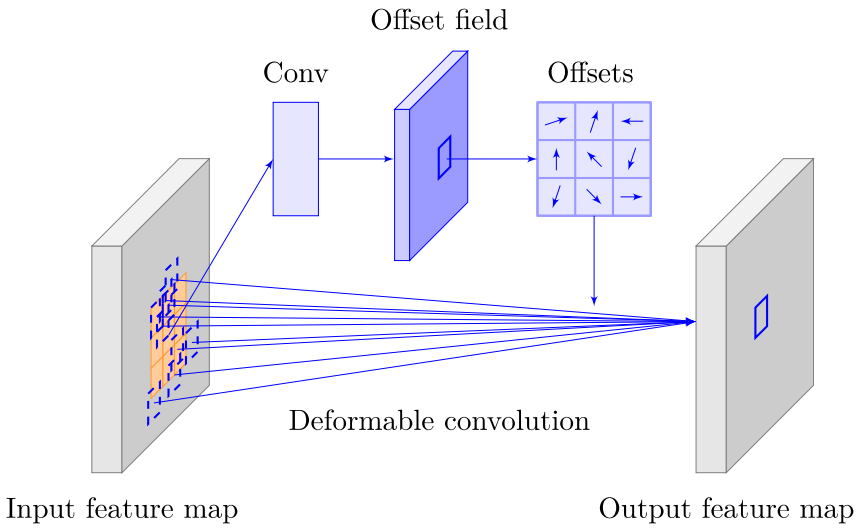


Figure 2.12: Illustration of parallel offset branch for 3x3 deformable convolution.

Compared to the previously mentioned equation for regular convolution 2.8, our deformable convolution equation now takes the form:

$$S(i,j) = \sum_m \sum_n I(i+m+\delta_m, j+n+\delta_n) F(m,n) \tag{2.9}$$

Where δ_m and δ_n are the offsets. Note that there is one (δ_m, δ_n) offset pair per filter position per feature map position. The offset locations are usually fractional and are computed by bilinear interpolation, which is differentiable. Both the offset layer and regular convolutional layer are trained simultaneously by backpropagating the gradient in equation (2.9).

2.3.3 Deformable RoI-pooling

RoI-pooling does course spatial quantization by converting an arbitrary size rectangular region into fixed size features. It does so by dividing the rectangular region into $k \times k$ bins by performing max-pooling. The region proposal rectangular region contains a lot of background information that is not related to the foreground object of interest. By adding learnable offsets to the bin locations we can move the bins to nearby object foreground regions. Deformable RoI-pooling adds offset parameters for the bins, see figure 2.13 for a comparison to regular RoI-pooling.

Figure 2.13: The left image illustrates standard RoI-pooling with a regular grid of 3×3 bins. The right image shows the result of deformable RoI-pooling with learnable offsets. The bins are adjusted to better cover the foreground object.

The deformable RoI-pooling module can replace regular RoI-pooling by adding an offset branch similar to how it is done in the deformable convolution module. However, instead of adding a convolutional layer we use a fully connected layer to compute the offsets. See figure 2.14 for an illustration of the deformable RoI pooling parallel branch.

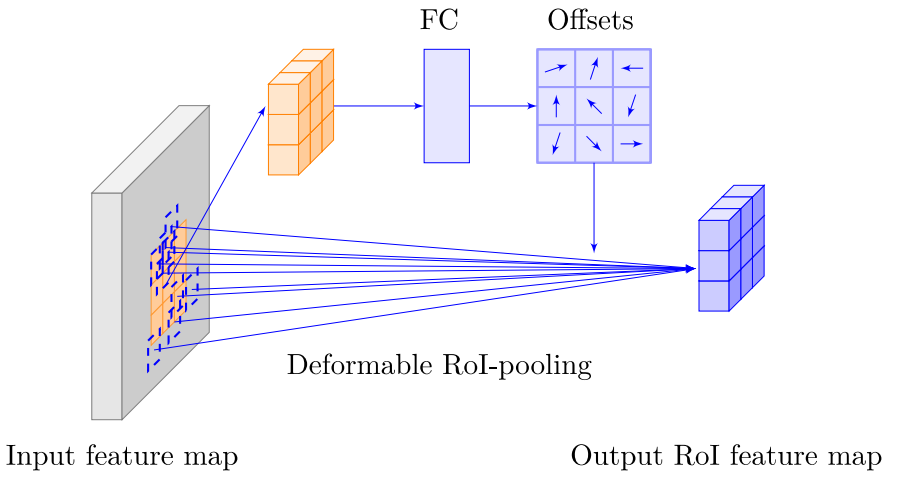


Figure 2.14: Illustration of parallel offset branch for 3x3 deformable RoI pooling.

2.4 Datasets

Some of the most common benchmark datasets that are used for object detection are PASCAL VOC[7], ImageNet[2] and COCO[13]. In this thesis we have performed experiments on two datasets, namely a small dataset of maritime images and PASCAL VOC. In this section we describe these two datasets.

2.4.1 Maritime Images

The Maritime Images dataset contains 1500 images of boats from two marinas in Gothenburg, namely Långedrag marina and Skeppstadsholmen marina. The images contains 5810 instances of approximately 300 unique boats of different kinds such as sailboats, motorboats and skiffs. The dataset does not contain any large passenger ships or freighters. All the boats have been annotated with the class label boat and there are no other classes in the dataset.
The bounding boxes for the boats have been annotated to be as tight as possible while still containing all the objects that are part of the boat. The objects that are considered to be part of the boat is the boat itself and everything that is attached to the boat such as masts, engines, anchors, fenders, boat covers, rails as well as antennas and other electronic equipment.
The images were taken by an Iphone 6s camera on a boat while driving inside the marinas. The width and height of the images are 3024 and 4032 pixels. The images were collected on a cloudy afternoon in May. No postprocessing has been done.
A sample subset of images with ground truth bounding boxes are shown in figure 2.15. As can be seen in the images, not all boats are annotated. The boats that are near unrecognizable due to occlusion or small size are not included. Furthermore, examples where only the mast is visible has not been annotated.
The dataset is split into a training, validation and test set containing 1000, 250 and 250 images respectively. The images are split such that no unique boat occurs in more than one set.

2.4.2 PASCAL VOC

The PASCAL Visual Object Challenge[7] (VOC) was an annual event from 2006 until 2012. It consisted of three primary challenges, namely object classification, object detection and semantic segmentation. The dataset of the challenges are publicly available together with ground truth annotations. A sample subset of the images in PASCAL VOC with ground truth bounding boxes are shown in figure 2.16.
The classes of the dataset comprises a person class as well as classes of different types of vehicles, household objects and animals, see table 2.1.
The datasets are divided into a training, validation and test set. Table 2.2 shows the number of images in the training, validation and test set for the 2007 and 2012 challenges. The union of the training set and validation set is referred to as the "trainval" set.

Figure 2.15: Sample images taken from the maritime images dataset with ground truth bounding boxes.

Vehicles	Household Objects	Animals	Other
Aeroplane	Bottle	Bird	Person
Bicycle	Chair	Cat	
Boat	Dining table	Cow	
Bus	Potted plant	Dog	
Car	Sofa	Horse	
Motorbike	TV/Monitor	Sheep	
Train			

Table 2.1: PASCAL VOC classes.

	Training Images	Validation Images	Test Images
VOC 2007	2,501	2,510	4,952
VOC 2012	5,717	5,823	10,991

Table 2.2: Number of images in the training, validation and test sets for the PASCAL VOC 2007 and PASCAL VOC 2012 challenges.

Figure 2.16: Sample images taken from the PASCAL VOC dataset with ground truth bounding boxes.

Chapter 3

Methods

This section explains the methodology used for this project. First, we briefly describe the implementation framework that has been used and the hardware that was used for the experiments. Secondly, we describe the model for the experiments that have been conducted on the dataset with maritime images as well as the model for the experiments where we combined deformable convolutions with other techniques that affect the spatial sampling locations of the convolutional filters. Finally, we go through the evaluation metric "mean average precision" that has been used to analyse the performance of the models.

3.1 Implementation Framework and Hardware

All the models have been implemented in the MXNet[1] framework and are based on the official implementation from the authors of the deformable convolutional network paper[2]. Apache MXNet is an open source deep learning framework that supports both imperative and symbolic programming as well as a wide range of programming languages.
The experiments in this thesis has been performed on a desktop computer with the following hardware specifications

- GPU: Single GeForce GTX 1070 8GB
- CPU: Intel I5-2500K CPU 3.30GHz
- RAM: 8GB 1600MHz

3.2 Maritime Images Experiments Model

In this section we present a baseline model for the experiments on the maritime images. Secondly, we describe the techniques that have been used to improve the baseline model. We end the section with an explanation of how the models have been trained.

[1] https://mxnet.apache.org/
[2] https://github.com/msracver/Deformable-ConvNets

3.2.1 Baseline Model

We have used the Faster R-CNN as a baseline model. The reason that we use the Faster R-CNN framework is that it is one of the most accurate detectors. Another reason that we use Faster R-CNN is that it is two-stage detector which allows us to replace the RoI-pooling layer with its deformable counterpart when we try to improve the baseline. The feature extractor that we use is a ResNet-101 convolutional network that has been pre-trained on the ImageNet dataset. The suffix indicates the number of layers in the residual network, i.e. 101 layers. There are deeper models that may perform better but are slower to train and more memory intensive. All layers in the baseline model uses regular convolutions and regular RoI-pooling. The model has been trained with 1 epoch with a warmup learning rate $5 * 10^{-5}$. Warmup is a training strategy that uses a lower learning rate in the beginning of training, which has been proven to be effective for avoiding early optimization difficulties[39]. After the warmup we have trained the model for 11 epochs with a learning rate of $5 * 10^{-4}$ and finally for 6 epochs with a learning rate of $5 * 10^{-5}$. The learning rate schedule has been determined by looking at the error on the validation set. In the training and testing phase, the images have been resized to have a shorter side of 600 pixels.

3.2.2 Improvements

In order to improve the baseline model we have used the following techniques, deformable layers, online hard example mining, data augmentation, soft-NMS and multi scale testing. The techniques have been applied incrementally in the order they are listed. Each of the techniques are explained below.

Deformable Layers

Deformable convolutions and deformable RoI-pooling is explained in section 2.3.
In our experiments, we have used deformable RoI-pooling and have replaced the last three convolutional layers in the ResNet feature extractor with their deformable counterpart.

Data Augmentation

Almost all computer vision tasks benefit from more data. Collecting data can be expensive and another way to increase the amount of training data is to create synthetic examples. One of the simplest methods to create synthetic data is to flip images horizontally. For object detection, many objects preserve their class when this operation is performed and there is no degradation of the bounding box positioning in relation to the object. With some augmentation techniques, such as slight rotation and random cropping, we risk either degrading the bounding box or changing the class. In the maritime images dataset, there are several objects that are tall but narrow, such as sailboats. If we rotate the sailboat, we risk degrading the bounding box by making it too small or too large for the object. If we apply random cropping, we might crop the image such that we can only see a piece of

electronic equipment of a boat. However, the electronic equipment should not be labeled as a boat unless it is attached to one. Therefore, care needs to be taken when cropping images in order to preserve the class of the objects.
For our experiments, we have only used horizontal flipping as data augmentation.

Online Hard Example Mining

One of the main differences between image classification and object detection is that in object detection we have a large class imbalance between foreground and background objects. This is especially true for single-stage detectors, but is also an issue for region proposal detection systems. One of the techniques that have been used historically to deal with this problem is called "bootstrapping"[40], which allows us to find useful negative background examples. When using bootstrapping, we first train a detector to recognize a certain object. Secondly, we freeze the model and try to detect objects in images that doesn't contain the object of interest in order to produce hard negatives. The hard negatives are then used again in the training phase to improve the detection accuracy. These two steps of training with the difficult examples and then using the model to find new hard negatives are repeated until a satisfactory accuracy has been reached.
One of the reasons that bootstrapping has not been used for convolutional networks is that freezing the model to find hard negatives considerably slows down the training. Online hard example mining (OHEM)[10] is a technique that performs a form of bootstrapping without introducing a significant computational overhead. In the Faster R-CNN object detection framework, OHEM works by increasing the number of region proposals in the forward pass and then does a hard example selection to pick out examples for which we should compute the gradient and use in the backward pass when we update the weights.
In our experiments we use OHEM and select 128 regions in the hard example selection process.

Soft-NMS

Almost all object detection systems use a post-processing technique called non-maximum suppression (NMS) in order to remove duplicate predictions of the same object. It works by choosing the prediction with the maximum confidence score and then removing the other predictions that have a significant overlap with that prediction. The maximum prediction is added to a list of final predictions and then the process is repeated again until all predictions have been processed. Figure 3.1 shows an example of applying the NMS technique.
The issue with standard NMS is that there may be two distinct objects with bounding boxes that have a significant overlap, which will cause NMS to remove the overlapping prediction with the lowest confidence score. Soft-NMS[41] addresses this problem by decreasing the confidence score of predictions that have an overlap with the prediction with maximum confidence score. The confidence score is decreased in proportion to the intersection over union overlap.
In our experiments we have replaced NMS with Soft-NMS for the final output predictions of the model.

(a) (b)

Figure 3.1: An example of non-maximum suppression. Image (a) shows the predictions of the object detection system before the non-maximum suppression technique has been applied. The thickness of the bounding boxes indicates the confidence score. Image (b) shows result after non-maximum suppression, where overlapping predictions have been removed.

Multi-Scale Testing

The objects in the maritime images dataset are composed of different structures at different scales. Multi-scale testing is a technique where we predict bounding boxes on an image at multiple scales. By predicting at multiple scales we can combine the information from both courser and finer features to improve the final accuracy.

We have used multi-scale testing at three different scales, such that the shorter side of the image is 480, 576 and 688 pixels.

3.3 Deformable Convolutions Experiments Model

In this section we present a baseline model and the architectural changes that have been proposed to that model. Furthermore, we describe how the models have been trained.

3.3.1 Baseline Model

Similarly to the maritime images experiments, we use a Faster R-CNN as a baseline model. The feature extractor that we use is a ResNet-101 convolutional network that has been pre-trained on ImageNet. The last three layers of the ResNet-101 convolutional network use deformable convolutions and the RoI-pooling layer uses deformable RoI-pooling.

3.3.2 Atrous Spatial Pyramid Pooling

Atrous spatial pyramid pooling (ASPP)[42] combines the ideas of atrous convolution and spatial pyramid pooling. In order to understand ASPP we first review these two concepts.
In regular convolutional filters we have that the spatial sampling locations are one spatial position apart. Atrous convolution is a type of convolution where the spatial sampling locations are set apart by a dilation rate d, which gives us an equation for two-dimensional convolution of the following form:

$$S(i,j) = \sum_m \sum_n I(i + d \times m, j + d \times n) F(m,n) \tag{3.1}$$

Atrous convolution is also known as dilated convolution. Figure 3.2 illustrates the difference between regular convolution and atrous convolution with a dilation rate of 3.

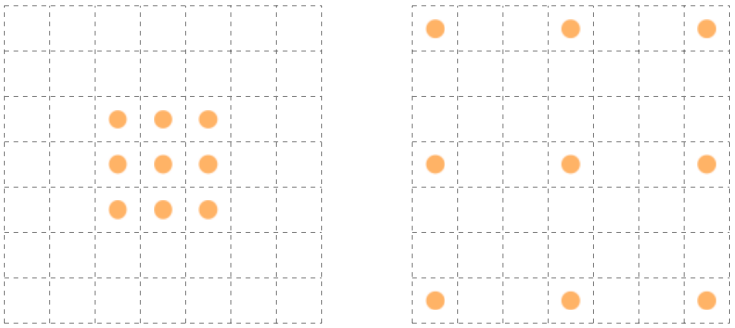


Figure 3.2: Illustration of the spatial sampling locations in a convolutional 3×3 filter. The left image illustrates the sampling locations of a regular convolutional filter and the right image illustrates a dilated filter with a dilation rate of 3.

Spatial pyramid pooling is a technique that was used in the convolutional network SPP-net[43]. It replaces a regular pooling layer with multiple pooling layers of different scales which are then concatenated together. This gives us an aggregation of coarser and finer features.
ASPP was proposed in the DeepLab[42] convolutional network for semantic segmentation. Instead of multiple pooling layers it uses convolutional filters with multiple dilation rates. The output feature maps are concatenated and then passed through a 1×1 convolutional layer, see figure 3.3.
In this thesis we have combined the techniques of deformable convolutions with ASPP in two ways. Firstly, we have applied ASPP to the entire deformable convolutional layer. Secondly, we have experimented with ASPP for just the offset branch.

ASPP for Entire Deformable Convolutional Layer

A deformable convolutional layer consists of two parts, an offset branch and a main branch. When applying ASPP to an entire deformable convolutional layer we have

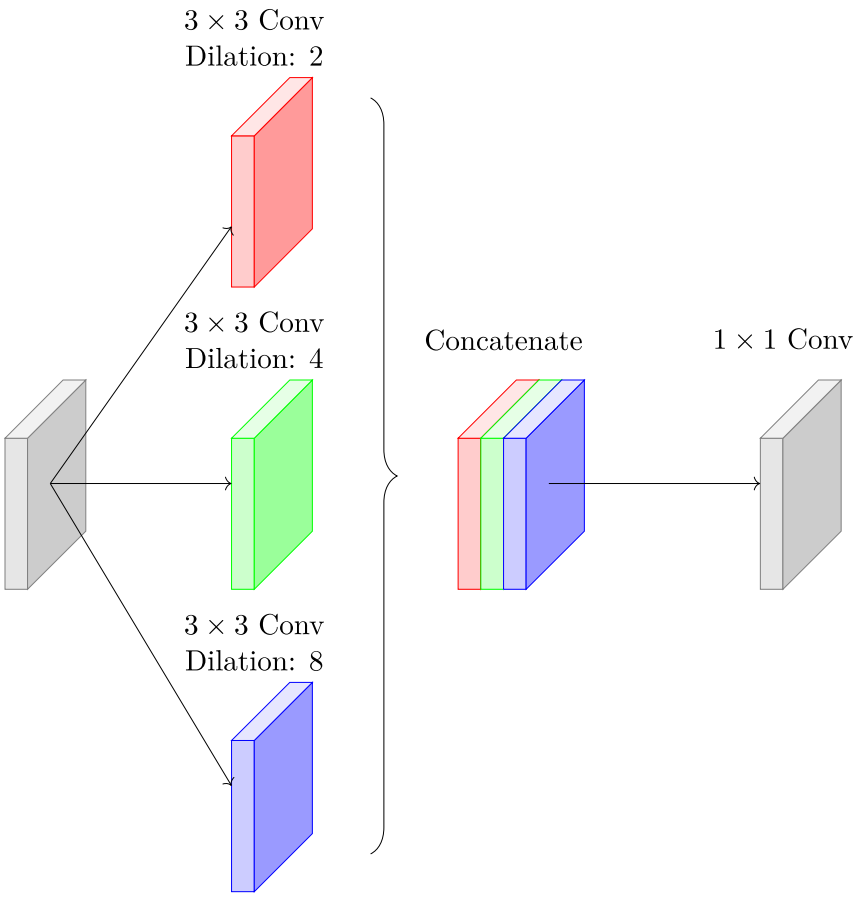


Figure 3.3: Example of atrous spatial pyramid pooling. The feature maps from three convolutional layers with different dilation rates are concatenated and finally passed through a 1×1 convolutional layer.

replicated the offset branch and main branch for each level of the pyramid. Because of the computational overhead, we have only used ASPP for the last deformable layer in the feature extraction network. The original deformable layer uses 512 filters in the main branch and 72 filters in the offset branch. Similarly, we have used 512 filters for the main branch and 72 offset filters for each level of the pyramid. The final 1×1 convolutional layer has 512 filters in order to keep the output dimension the same as in the original model. In our experiments we have tried ASPP with two different dilation rates $(2, 4)$ and three different dilation rates $(2, 4, 8)$. The offset branch uses the same dilation rate as the main branch it is connected with.

ASPP for Deformable Offset Branch

Figure 3.4 illustrates the structural changes when applying ASPP to the offset branch. Since applying ASPP to the offset branch is computationally less expensive than applying it to an entire deformable layer, we have applied ASPP to the offset branch of all deformable layers. We have used the same number of filters for each level of the pyramid as in the original offset branch, namely 72 filters. The final 1×1 convolutional layer has 72 filters as well. We have used the same number of different

dilation rates as in the other other ASPP experiment, i.e. $(2, 4)$ and $(2, 4, 8)$. The main branch of the deformable layer uses the original dilation rate of 2.

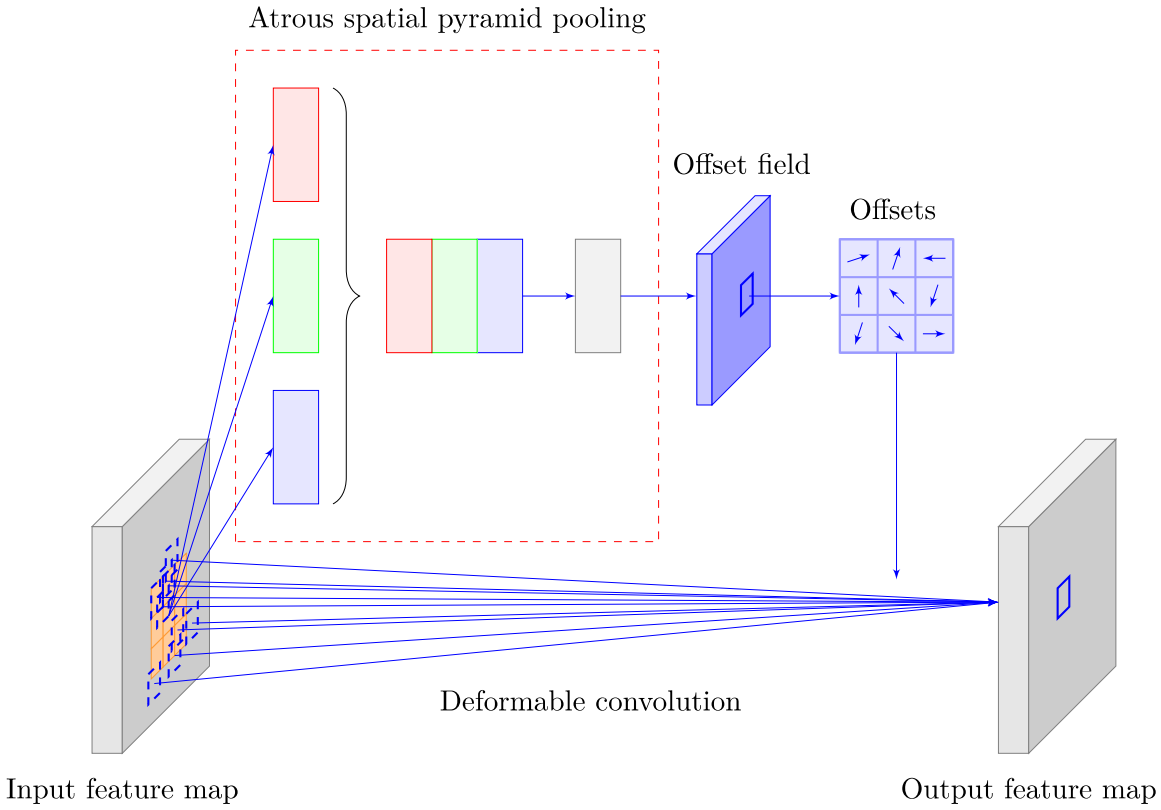


Figure 3.4: Example of atrous spatial pyramid pooling with three different dilation rates for the parallel offset branch of a deformable convolutional layer.

Elementwise Adding

For both of the experiments above we have also substituted the concatenation step with another fusion strategy, namely elementwise adding. We use the same number of filters for all convolutions as in the experiments above. However, the final 1×1 convolutional is no longer needed since the output dimension is already correct after the elementwise adding.

3.3.3 Multiple Filter Size

In the ASPP experiment we used several branches with different dilation rates and then fused the results with either concatenation or elementwise summation. Drawing inspiration from the Inception-block in the convolutional architecture of Szegedy et. al.[44], we have also tried branches with different filter sizes in the final layer. Similarly to the ASPP experiment we have used the fusion strategies of concatenation and elementwise summation. The original kernel size of the last layer is 3×3. In our experiments we have tried the following combinations of branches with different

kernel sizes: $(3 \times 3, 5 \times 5), (1 \times 1, 3 \times 3, 5 \times 5)$. Each branch with different filter size comprises 512 filters. Furthermore, we have conducted experiments with a single kernel size of 3×3 and 5×5 for reference. All experiments have been run for the deformable model as well as its non-deformable counterpart.

3.3.4 Training

The training parameters are for the most part similar to the default configuration for the Faster R-CNN version of the official implementation of deformable convolutional networks. The training dataset is the union of the PASCAL VOC 2007 trainval and 2012 trainval datasets and the images are resized to have a shorter side of 600 pixels. The reason that we have chosen to perform these experiments on PASCAL VOC instead of the maritime images dataset or the larger COCO dataset is that PASCAL VOC contains enough images to give reliable results but is small enough to converge after a reasonable amount of time given the computational resources available for this project. The models have been trained for 7 epochs, where one epoch is one pass through the entire training data.

We use the same learning rate scheme as the official implementation. First we use a warmup learning rate of $5 * 10^{-5}$ for 4000 iterations. After the warmup stage, the learning rate is set to $5 * 10^{-4}$ and $5 * 10^{-5}$ in the first $\frac{2}{3}$ and last $\frac{1}{3}$ iterations, respectively.

Due to running into issues with exploding gradients during training we have set a gradient clipping value of 1. Gradient clipping limits the magnitude of the gradients to protect against gradients from extreme cases, which can have a devastating effect on the weights.

3.4 Evaluation

The evaluation metric that has been used is the mean average precision (mAP) score as defined by PASCAL VOC[7]. The deformable convolutions experiments have been evaluated on the PASCAL VOC 2007 test set. The improvements on the maritime images experiments have been evaluated on the maritime images validation set and then the final model has been evaluated once on the maritime images test set.

3.4.1 Mean Average Precision

The output of our model is a list of predictions where each prediction consists of coordinates for the bounding box, a class and a confidence score. The confidence score indicates how certain the model is that the bounding box contains the object of interest.

The mean average precision is obtained by calculating the mean of the average precision (AP) of all classes. In order to understand the average precision metric, we first review the concepts of precision and recall. Precision and recall are computed

in the following way

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{3.2}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{3.3}$$

In the case of object detection, a positive prediction is when the confidence score of the prediction exceeds a confidence threshold t. The prediction is a true positive if the overlap ratio between the predicted bounding box and a ground truth bounding box with the correct class exceeds an intersection over union (IoU) value v. The prediction is a false positive if there is no ground truth bounding box with the correct class that has an overlap ratio with the predicted bounding box which exceeds the intersection over union value v. If there are multiple positive predictions satisfying the overlap criterion for a certain ground truth object, the first prediction is counted as a true positive and the other predictions are counted as false positives. A false negative is when there is a ground truth objects which doesn't satisfy the overlap criterion with any positive prediction of the same class.
The intersection over union value is defined by the formula

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{3.4}$$

where B_p and B_{gt} are the predicted bounding box and ground truth bounding box, respectively.
There is a trade-off between precision and recall. With a higher confidence threshold we get a higher precision and with a lower confidence threshold we get a higher recall. By plotting the precision against the recall we get a precision/recall curve that shows the trade-off between precision and recall at all relevant confidence thresholds.
The average precision can be seen as a summary statistic of the precision/recall curve. It is computed by taking the mean of the precision p at eleven recall levels r

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p(r) \tag{3.5}$$

Chapter 4

Results

This chapter presents the results of the two types of experiments that have been conducted. First, we show the results of the experiments on the maritime images where the goal was to examine which performance could be achieved by applying a range of techniques. Secondly, we show the results of the experiments where we combined deformable convolutions with other techniques that affects the spatial sampling locations of the convolutional filters.

4.1 Maritime Images Experiments

In this section we show the results of the experiments on the maritime images dataset. We have trained a baseline model and then applied the following improvements, deformable layers, online hard example mining, horizontal flip, soft-NMS and multi-scale testing. The techniques have been applied incrementally in the order listed above and the implementation details are listed in section 3.2. Table 4.1 shows the mAP score at the intersection over union thresholds of 0.5 and 0.7 when the models have been tested on the maritime images validation set. We can see that the biggest improvement comes from adding the deformable RoI-pooling and deformable convolutional layers, especially for the mAP score at the IoU threshold of 0.7. Table 4.2 shows the mAP score at the intersection over union thresholds of 0.5 and 0.7 for the model with all improvements when tested on the maritime images test set. The score is a bit lower than the score on the validation set. This is expected since the hyperparameters such as learning rate and number of epochs have been tuned according to the performance on the validation set. A visual assessment of the detections is given in the discussion section 5.1
The techniques that improve the mAP score come at a cost in terms of training and test speed. Table 4.3 shows the training and test speed in images per second when applying the aforementioned techniques. The hardware that has been used is listed in section 3.1. As before, the techniques are applied incrementally. We note that using deformable layers has the most impact on the training speed out of all the listed techniques. However, it is also the technique that improves the mAP score the most. Multi-scale testing significantly decreases the test speed, almost by a factor equal to the number of different test scales. Worth noting is that the horizontal flip doesn't increase the training speed nor the test speed, but still gives a considerable

Maritime Images Validation Set	mAP@0.5	mAP@0.7
Baseline	78.0	60.6
+ Deformable layers	80.1	66.4
+ Online Hard Example Mining	81.1	66.5
+ Horizontal Flip	81.9	68.9
+ Soft-NMS	82.2	69.9
+ Multi-Scale Testing	**83.7**	**70.4**

Table 4.1: Detection mean average precision (mAP) at IoU thresholds 0.5 and 0.7 on the maritime images validation set. The table shows the result of applying a range of techniques incrementally, i.e. at the final row all techniques are applied.

Maritime Images Test Set	mAP@0.5	mAP@0.7
Final Model	79.9	57.9

Table 4.2: Detection mean average precision (mAP) at IoU thresholds 0.5 and 0.7 on the maritime images test set.

increase in mAP score.

Maritime Images	Training Speed (images/second)	Test Speed (images/second)
Baseline	3.89	10.5
+ Deformable layers	3.35	9.35
+ Online Hard Example Mining	3.31	9.35
+ Horizontal Flip	3.31	9.35
+ Soft-NMS	3.31	9.30
+ Multi-Scale Testing	3.31	3.09

Table 4.3: Training speed and test speed in images per second on the maritime images. The techniques listed in the table are applied to the baseline model incrementally.

Figure 4.1 shows a sample of maritime images from the test set with detections generated by the final model. Detections with a confidence score higher than 0.7 are shown.

4.2 Deformable Convolutions Experiments

In this section we show the results of applying atrous spatial pyramid pooling and an Inception-like layer of multiple filter sizes to a model with deformable convolutions. All results are shown with an intersection over union threshold of 0.5 and 0.7. An analysis of the results is given in the discussion section 5.2.

Figure 4.1: Sample maritime images from the test set with detections generated by the final model.

4.2.1 Atrous Spatial Pyramid Pooling

Two types of atrous spatial pyramid pooling (ASPP) experiments have been performed. ASPP on the entire deformable convolutional layer and ASPP on the offset branch exclusively. See section 3.3.2 for a detailed description.

ASPP on Entire Deformable Convolutional Layer

Table 4.4 shows the result of adding ASPP to the last convolutional layer in the feature extraction network. We compare the results of using ASPP on a regular convolutional network and a deformable convolutional network. We can see that the mAP score of the deformable models with ASPP is similar to the mAP score of the non-ASPP deformable model. However, ASPP offers a slight improvement in mAP score at both IoU thresholds for the models with regular convolution.

ASPP Last Convolutional Layer	**mAP@0.5**	**mAP@0.7**
Regular Conv Baseline	78.0	60.4
Regular Conv + Concat (2,4)	**78.6**	61.0
Regular Conv + Concat (2,4,8)	78.4	61.2
Regular Conv + Add (2,4)	78.3	61.2
Regular Conv + Add (2,4,8)	78.3	**62.0**
Deformable Conv Baseline	80.0	66.7
Deformable Conv + Concat (2,4)	**80.2**	66.5
Deformable Conv + Concat (2,4,8)	80.0	66.5
Deformable Conv + Add (2,4)	79.7	**66.9**
Deformable Conv + Add (2,4,8)	80.1	66.6

Table 4.4: Detection mean average precision (mAP) at IoU thresholds 0.5 and 0.7 on VOC 2007 test. The table shows the result of applying atrous spatial pyramid pooling to the last convolutional layer when regular convolution and deformable convolution is used. The table shows the result for two different dilation rates (2,4) as well as three different dilation rates (2,4,8). Two different fusion strategies have been used, concatenation with 1×1 convolution (Concat) and elementwise adding (Add).

ASPP on Deformable Offset Branch

Table 4.5 shows the result of adding ASPP to the offset layer of all three deformable convolution layers in the feature extraction network. From the table we can see that the ASPP with elementwise adding gives slightly better results at the IoU threshold of 0.5, but slightly worse results at the IoU threshold of 0.7. As for the other fusion strategy of concatenation, we have a decrease in performance for both IoU thresholds.

ASPP Offset Layers	mAP@0.5	mAP@0.7
Deformable Conv Baseline	80.0	**66.7**
Deformable Conv + Offset Concat (2,4)	79.8	65.9
Deformable Conv + Offset Concat (2,4,8)	79.4	65.6
Deformable Conv + Offset Add (2,4)	**80.4**	66.3
Deformable Conv + Offset Add (2,4,8)	80.3	66.6

Table 4.5: Detection mean average precision (mAP) at IoU thresholds 0.5 and 0.7 on VOC 2007 test. The table shows the result of applying atrous spatial pyramid pooling to the offset layer in all three deformable convolutional layers. The table shows the result for two different dilation rates (2,4) as well as three different dilation rates (2,4,8). Two different fusion strategies have been used, concatenation with 1×1 convolution (Concat) and elementwise adding (Add).

4.2.2 Multiple Filter Size

Table 4.6 shows the result of modifying the last convolutional layer in the feature extraction network to be Inception-like with multiple filter sizes. We compare the results of using the modified layer on a regular convolutional network and a deformable convolutional network. We can see that the mAP score of the deformable models with the modified layer with multiple filter sizes is slightly worse than the mAP score of the original deformable model. There is a single configuration "Deformable Conv + Concat$(3 \times 3, 5 \times 5)$" for which the baseline is beaten at the IoU threshold of 0.5, the score is however slightly worse at the IoU threshold of 0.7. The modified layer with multiple filter sizes offers a slight improvement in mAP score at both IoU thresholds for the models with regular convolution.

Multiple Filter Sizes	**mAP@0.5**	**mAP@0.7**
Regular Conv (3×3)	78.0	60.4
Regular Conv (5×5)	78.3	61.0
Regular Conv + Concat ($3 \times 3, 5 \times 5$)	78.3	61.2
Regular Conv + Concat ($1 \times 1, 3 \times 3, 5 \times 5$)	78.3	61.0
Regular Conv + Add ($3 \times 3, 5 \times 5$)	78.4	61.1
Regular Conv + Add ($1 \times 1, 3 \times 3, 5 \times 5$)	**78.5**	**61.4**
Deformable Conv (3×3)	80.0	**66.7**
Deformable Conv (5×5)	79.9	66.4
Deformable Conv + Concat ($3 \times 3, 5 \times 5$)	**80.4**	66.6
Deformable Conv + Concat ($1 \times 1, 3 \times 3, 5 \times 5$)	79.9	66.3
Deformable Conv + Add ($3 \times 3, 5 \times 5$)	79.7	**66.6**
Deformable Conv + Add ($1 \times 1, 3 \times 3, 5 \times 5$)	80.0	66.5

Table 4.6: Detection mean average precision (mAP) at IoU thresholds 0.5 and 0.7 on VOC 2007 test. The table shows the result of using multiple filter sizes on the last convolutional layer when regular convolution and deformable convolution is used. The table shows the result for two different combinations of filter sizes $(3 \times 3, 5 \times 5), (1 \times 1, 3 \times 3, 5 \times 5)$. Furthermore, the results of experiments with two different single filter sizes of 3×3 and 5×5 are also shown. Two different fusion strategies have been used, concatenation with 1×1 convolution (Concat) and elementwise adding (Add).

Chapter 5

Discussion

In this chapter, we first do a visual assessment of the detections on the maritime images test set. Secondly, we analyze the results of the deformable convolution experiments and try to understand why some techniques work with regular convolutional networks, but not with deformable convolutional networks. We conclude the chapter with a further work section.

5.1 Visual Assessment of Detections on Maritime Images

In this section we visually assess the detections on the maritime images test set to see what kind of errors the model makes. For all the images in this section, we have chosen to include detections with a confidence score above 0.7. This threshold has been chosen to balance between false positives and false negatives. We start by looking at images where the model has failed to detect boats with a ground truth label. Two of the main reasons that a boat is not detected by our model are occlusion and clutter. Figure 5.1 shows a few images where ground truth objects does not have a matching detection due to occlusion and clutter. For cluttered objects, the model may consider the entire collection of cluttered objects as a single instance, with a bounding box encompassing all of the cluttered objects. However, in some instances the model just detects one of the cluttered objects. Distinguishing cluttered objects of the same class with duplicate detections of the same object is a general problem that object detection systems that use non-maximum suppression struggles with.
A second type of error that our model makes is that it detects boats that are not boats at all. Figure 5.2 shows a few images illustrating this kind of error. Two of the false positives seem to have a mast-like appearance, which may explain why the model may have thought that they are boats. If this kind of error is considered more serious than a potential increase in false negatives, then the confidence threshold can be increased.
Although some of the false positives are not boats at all, most of the false positives are actual boats that have not been annotated as boats due to the criterias listed in section 2.4.1. Figure 5.3 shows a few examples of where the model detected boats that do not have a matching ground truth object. We can see that most of

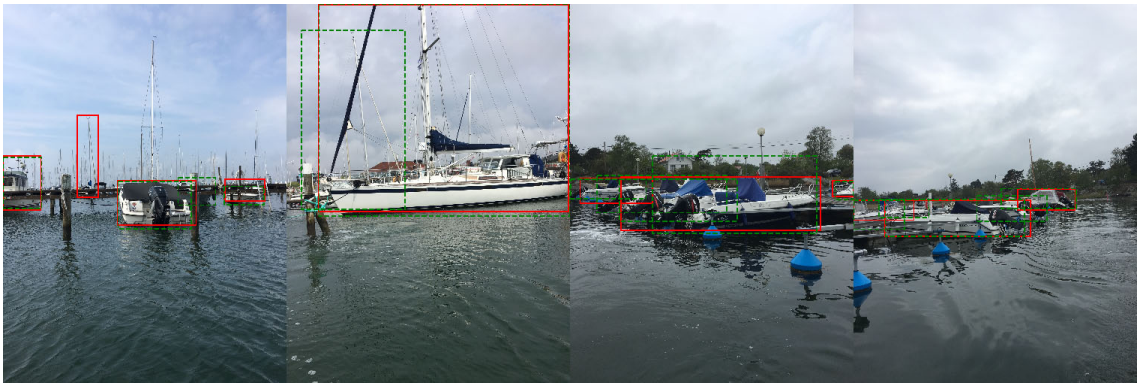


Figure 5.1: Images with missing or inaccurate detections due to occlusion and clutter. Detections are shown as solid red rectangles and ground truth objects as dashed green rectangles.

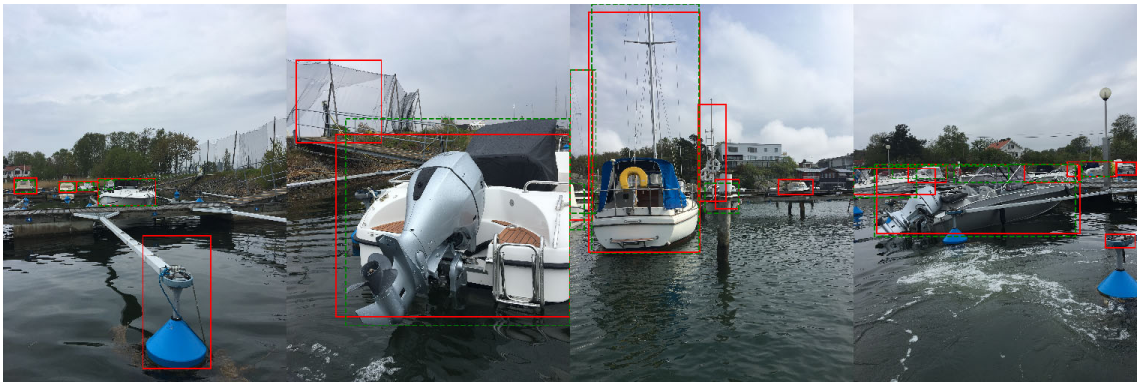


Figure 5.2: Images with detections of non-boats as boats. Detections are shown as solid red rectangles and ground truth objects as dashed green rectangles.

the examples are boats in the background and are examples that almost meet the criteria of being in the annotation set. We can also see an example of a rubber boat that is on the back of a sailboat. When the rubber boat is attached to the boat, it is considered to be part of the boat and not an instance in itself. However, when the rubber boat is floating in the water, then it has been labeled as an individual object. There are just a few examples with these conditions and the model has not learned these fine nuances.

Except for the problems that are mentioned above, the model makes a few other mistakes. The model has learned quite well to match the masts with the boats. However, there are a few instances where the model either does not include the mast or the model returns one prediction for the boat excluding the mast and one prediction for the boat including the mast. Figure 5.4 illustrates the problem of duplicate detections for sailboats.

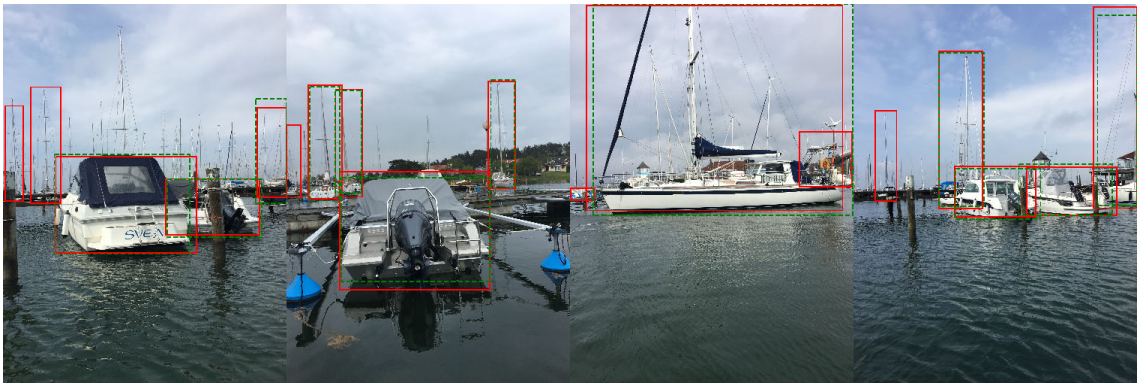


Figure 5.3: Images with detections of boats that are not in the annotation set. Detections are shown as solid red rectangles and ground truth objects as dashed green rectangles.

Figure 5.4: Images with duplicate detections of sailboats. Detections are shown as solid red rectangles and ground truth objects as dashed green rectangles.

5.2 Analysis of the Results of Deformable Convolution Experiments

From the results we can see that using multiple dilation rates and using multiple filter sizes in the last layer of a regular convolutional network improved the performance. However, neither techniques had a significant improvement on the convolutional network with deformable layers. To understand why this might be, we study the spatial sampling locations of the last three layers for a particular position in an image of the PASCAL VOC 2007 test set. The intuition behind using multiple dilation rates and multiple filter sizes is that we would like to capture information that is distributed in a small area with convolutions that have a small dilation rate or small filter size and information that is distributed in a larger area with convolutions that have a larger dilation rate or larger filter size. Figure 5.5 shows the sampling locations when using dilation rates 2, 4 and 8 in the last convolutional layer in both a regular and deformable convolutional network. It is hard to establish why the multiple dilation rates don't work as well for the deformable convolutional network, but one theory is that the deformable convolutional layer with dilation rate 2 already adapts quite well to the object of interest making the other deformable

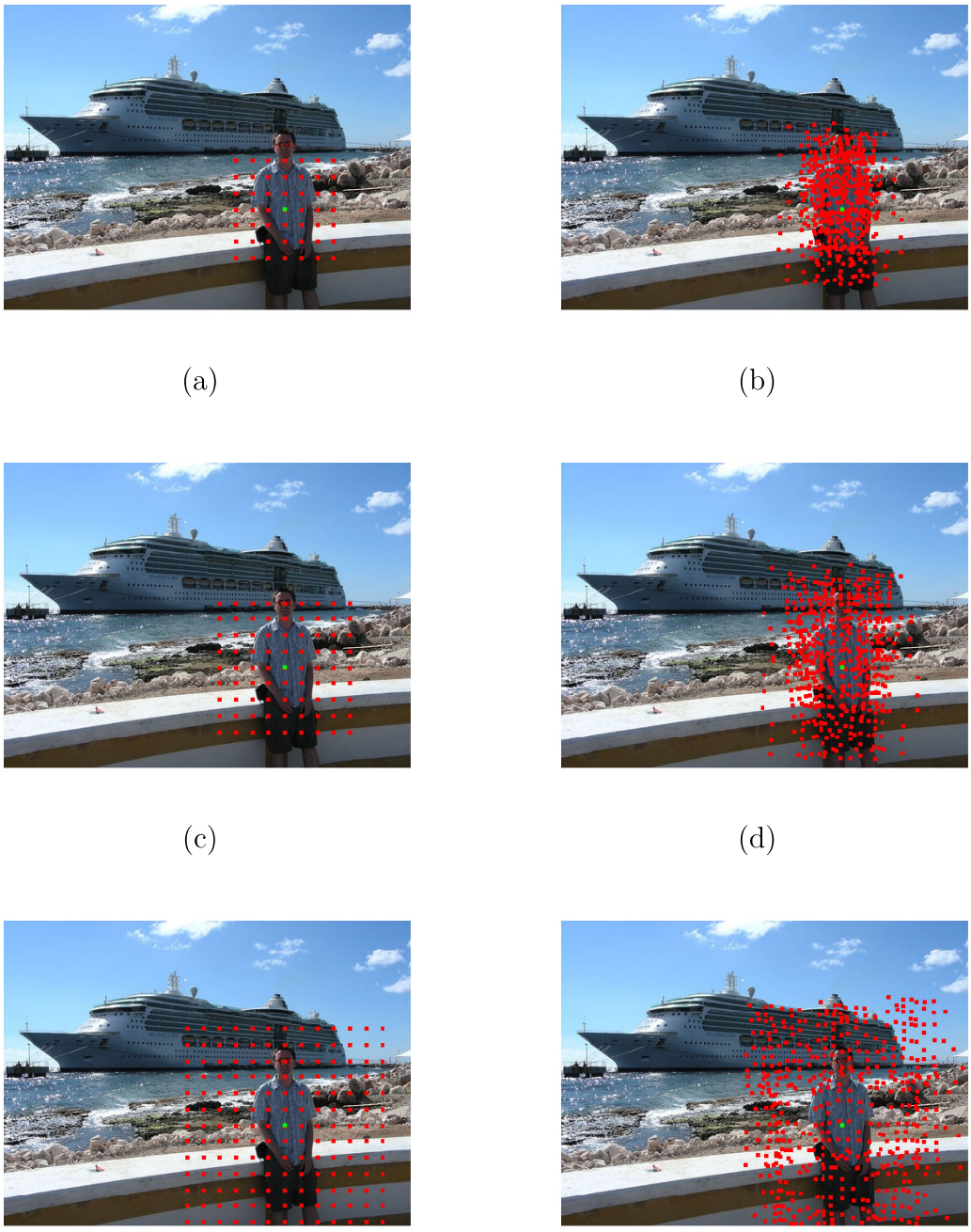
convolutional layers with other dilation rates superfluous.
Figure 5.6 shows the sampling locations when using filter sizes 3×3 and 5×5 in the last convolutional layer in both a regular and deformable convolutional network. Similarly to the multiple dilation case, we can see that the deformable sampling locations already cover the object quite well for the filter with size 3×3. This might explain why we do not get a performance increase when adding or concatenating the outputs of convolutional layers with both filter sizes.

5.3 Further Work

In this section we present other techniques that can improve the mAP score of the network in the maritime images experiments. Furthermore, we give a suggestion of how to improve the deformable convolution operator which we call learnable dilation.

5.3.1 Other Techniques to Improve mAP Score

One of the main issues with the maritime images dataset is that it does not contain many examples. Gathering more data would probably the most effective measure to increase the mAP score. Another option is to use more data augmentation techniques. Except for the flipping, rotation and cropping mentioned in the methods section, we can add gaussian noise or use more advanced techniques such as Generative Adversial Networks[45] to create synthetic examples. Data augmentation is not limited to the training phase but can also used in the test phase similar to how we used multi-scale testing.
Changing the architecture is another way to improve the model. Deeper architectures generally perform better. In our experiments we used a ResNet feature extractors with 101 layer, but by increasing it to 152 layers we might see a small increase in performance. The performance gains of the depth comes at the cost of increased training speed and memory requirements. Changing the architecture entirely might also provide a boost in performance. DenseNet[46], ResNeXt[47], Xception[48] and Inception-ResNet[49] are popular alternatives to the ResNet and they also use residual connections. To improve the mAP score even further, we can combine these feature extractors with the feature pyramid network technique illustrated in section 2.2.3.
We can also use an ensemble of different models and average their predictions to increase performance. The idea behind ensembling is that the models make different types of mistakes, and that the majority of them should get their prediction right. In order for the models not to make the same kind of mistakes, we can use different architectures and train them on different folds of the training data. While an ensemble of models may perform better, it can take a long time to train them. Instead of averaging over the predictions, we can also average over the weights in a network. Stochastic weight averaging[50] takes the average of the weights in a single model at different iterations. Weight averaging leads to finding a wider minima and better generalization.
Object detection is an active research area. Other than the techniques mentioned above, there are several other ways to improve a model, and several more to come.

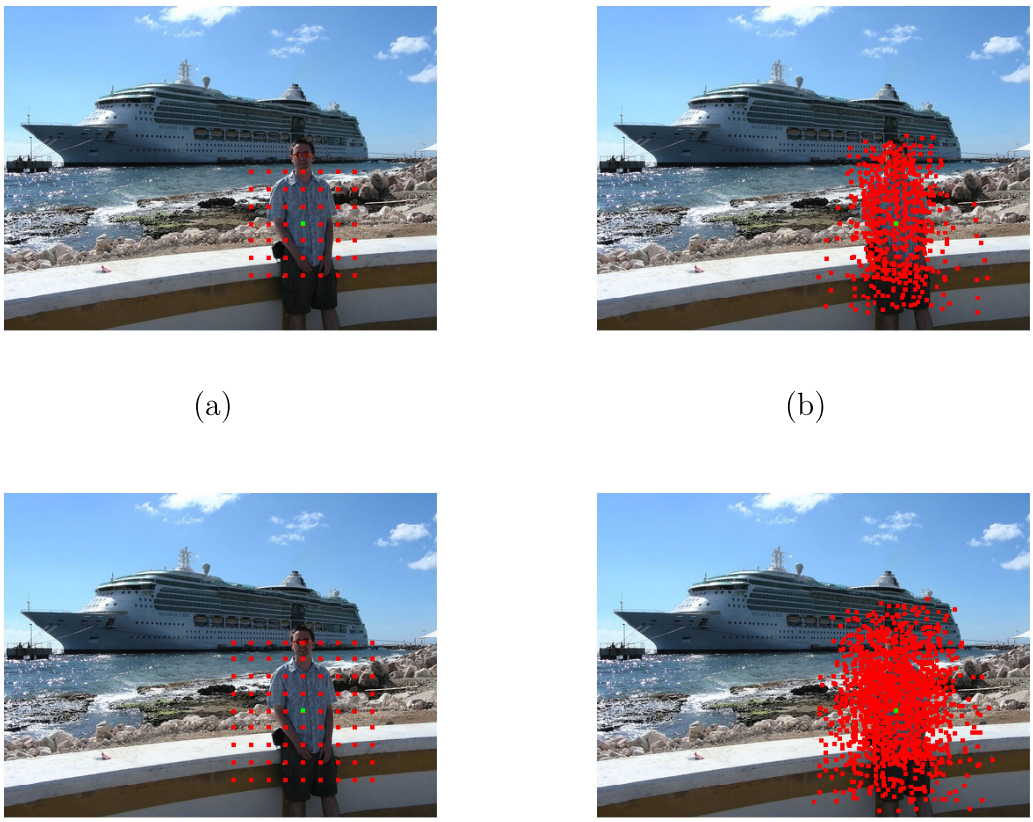


(a) (b)

(c) (d)

(e) (f)

Figure 5.5: The red dots show the sampling locations of the final three layers of a deformable convolutional network for the neuron located at the green dot. In the left images we have the spatial sampling locations for a network with regular convolutions where the dilation rate in the last layer is 2, 4 and 8 in subfigure (a), (c) and (e) respectively. In the right images we have the spatial sampling locations for a network with deformable convolutions where the dilation rate in the last layer is 2, 4 and 8 in subfigure (b), (d) and (f) respectively. The number of spatial sampling locations are the same in all images, but there is a significant overlap for the locations in the networks using regular convolutions.



(a) (b)

(c) (d)

Figure 5.6: The red dots show the sampling locations of the final three layers of a convolutional network for the neuron located at the green dot. In the left images we have the spatial sampling locations for a network with regular convolutions where the filter size in the last layer is 3 in subfigure (a) and 5 in subfigure (c). In the right images we have the spatial sampling locations for a network with deformable convolutions where the filter size in the last layer is 3 in subfigure (b) and 5 in subfigure (d).

5.3.2 Learnable Dilation

Deformable convolution adds an offset height and width parameter for each filter element for each spatial location. For a 3×3 filter this gives us $3 \times 3 \times 2$ parameters for each spatial location. An interesting modification of this would be to redesign the deformable convolution operator to learn a height and width dilation offset. That would give us 2 parameters per filter for each spatial location instead. Figure 5.7 illustrates the difference between regular deformable convolution and the modified learnable dilation version.

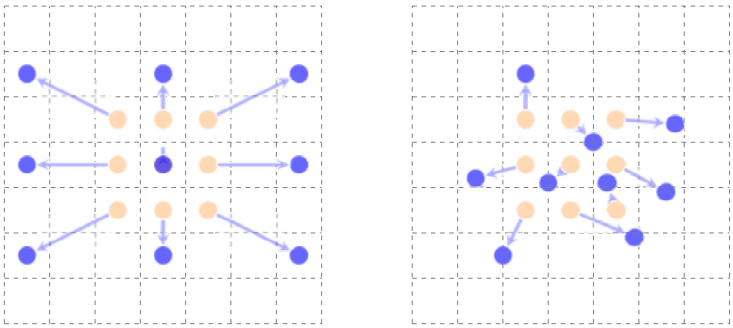


Figure 5.7: Illustration of the spatial sampling locations in a convolutional 3×3 filter. The left image illustrates the sampling locations of a learnable dilation filter with the dilation offset represented by the arrows. The right image illustrates a deformable filter with offsets represented by the arrows. Note that there are only two learnable parameters for the learnable dilation filter, namely the horizontal and vertical dilation rate. For the deformable convolution we have 2 parameters for each element of the filter.

In section 5.2 we looked at the spatial sampling locations of the last three layers of a deformable convolutional network. From the figures it can be seen that the offsets primarily learns the scale and proportions of the objects. Our hypothesis is that the modified learnable dilation version should give comparable accuracy while reducing the number of offset parameters as well as the training speed.

Chapter 6

Conclusion

In this thesis, we examined what accuracy we could achieve when detecting boats in small boat marinas. Furthermore, we have looked deeper into the interplay of deformable convolutional networks and other techniques that affect the spatial sampling locations of convolutional filters.
For the deformable convolution experiments, we examined the results of deformable convolutions with atrous spatial pyramid pooling and an inception-like block of multiple filter sizes. We compared the findings with the results of applying the atrous spatial pyramid pooling and the inception-like block of multiple filter sizes to a regular convolutional network without deformable convolutions. The results showed that the atrous spatial pyramid pooling and inception-like block gave a consistent increase in mAP score for the regular convolutional network. However, when combining atrous spatial pyramid pooling and the inception-like block with deformable convolutions, the mAP score did not improve. We conclude that deformable convolutions are not compatible with these techniques.
For the maritime images experiments, we have collected and annotated 1000 training images and evaluated them on 250 test images. The techniques that were employed and evaluated were deformable layers, data augmentation, online hard example mining, soft-NMS and multi-scale testing.
A visual assessment of the results showed that the model did well on boats that were clearly visible in the foreground. However, the model struggled with cluttered and occluded objects as well as objects far in the background. The mean average precision of the final model was 79.9 and 57.9 at the intersection over union thresholds 0.5 and 0.7 respectively. The results showed that adding deformable layers significantly improved the model with an increase in mean average precision of 5.8 at the intersection over union threshold 0.7.

Bibliography

[1] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *null*, page 511. IEEE, 2001.

[2] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[3] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE Computer Society, 2005.

[4] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

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

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

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

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

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

[10] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–769, 2016.

[11] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.

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

ceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2117–2125, 2017.

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

[14] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. *arXiv preprint arXiv:1901.01892*, 2019.

[15] Bharat Singh, Mahyar Najibi, and Larry S Davis. Sniper: Efficient multi-scale training. In *Advances in Neural Information Processing Systems*, pages 9310–9320, 2018.

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

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

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

[19] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[20] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[21] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[24] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118, 2010.

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

[28] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010.

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

[30] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. Megdet: A large mini-batch object detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6181–6189, 2018.

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

[32] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

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

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

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

[36] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. *arXiv preprint arXiv:1811.11168*, 2018.

[37] David G Lowe. Object recognition from local scale-invariant features. In *iccv*, page 1150. Ieee, 1999.

[38] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

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

[40] Kah K Sung and Tomaso Poggio. Example based learning for view-based human face detection. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 1994.

[41] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms–improving object detection with one line of code. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5561–5569, 2017.

[42] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018.

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

[44] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[45] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[46] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[47] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[48] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[49] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[50] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.