A Novel Animal Detection Technique for Intelligent Vehicles

by

Weihong Zhao

Thesis submitted in partial fulfillment of the requirements for the
Master of Applied Science degree in Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

The animal-vehicle collision has been a topic of concern for years, especially in North America. To mitigate the problem, this thesis focuses on animal detection based on the onboard camera for intelligent vehicles.

In the domain of image classification and object detection, the methods of shape matching and local feature crafting have reached the technical plateau for decades. The development of Convolutional Neural Network (CNN) brings a new breakthrough. The evolution of CNN architectures has dramatically improved the performance of image classification. Effective frameworks on object detection through CNN structures are thus boosted. Notably, the family of Region-based Convolutional Neural Networks (R-CNN) perform well by combining region proposal with CNN. In this thesis, we propose to apply a new region proposal method—Maximally Stable Extremal Regions (MSER) in Fast R-CNN to construct the animal detection framework.

MSER algorithm detects stable regions which are invariant to scale, rotation and viewpoint changes. We generate regions of interest by dealing with the result of MSER algorithm in two ways: by enclosing all the pixels from the resulted pixel-list with a minimum enclosing rectangle (the PL MSER) and by fitting the resulted elliptical region to an approximate box (the EL MSER). We then preprocess the bounding boxes of PL MSER and EL MSER to improve the recall of detection. The preprocessing steps consist of filtering out undesirable regions by aspect ratio model, clustering bounding boxes to merge the overlapping regions, modifying and then enlarging the regions to cover the entire animal. We evaluate the two region proposal methods by the measurement of recall over IoU-threshold curve. The proposed MSER method can cover the expected regions better than Edge Boxes and Region Proposal Network (RPN) in Faster R-CNN.

We apply the MSER region proposal method to the framework of R-CNN and Fast R-CNN. The experiments on the animal database with moose, deer, elk, and horses show that Fast R-CNN with MSER achieves better accuracy and faster speed than R-CNN with MSER. Concerning the two ways of MSER, the experimental results show that PL MSER is faster than EL MSER and EL MSER gains higher precision than PL MSER. Also, by altering the structure of network used in Fast R-CNN, we verify that network stacking more layers achieves higher accuracy and recall.

In addition, we compare the Fast R-CNN framework using MSER region proposal with the state-of-the-art Faster R-CNN by evaluating the experimental results of on our animal database. Using the same CNN structure, the proposed Fast R-CNN with MSER gains a higher average accuracy of the animal detection 0.73, compared to 0.42 of Faster R-CNN. In terms of detection quality, the proposed Fast R-CNN with MSER achieves better IoU histogram than that of Faster R-CNN.

Acknowledgments

I would like to express my sincere gratitude to my supervisor Professor Azzedine Boukerche for his financial support, warm encouragement and insistent guidance all through my master research. I am fortunate and proud to have him as my mentor. Whenever I ran into troubles, he would always offer help. He also helped me to build up my confidence.

In the meantime, I want to thank all my colleagues and friends in Paradise Lab. My mentor Abdelhamid Mammeri helped me a lot in the direction of my research and the writing of my thesis. Dr. Peng Sun offered me many constructive suggestions for my research and thesis. I also feel incredibly grateful to my friends Chong Luo, Zhijun Hou, and He Li, etc. for their friendship and encouragement.

I would like to give my special thanks to my husband. He supported me by fixing the problems with my hardware and sharing his high-performance computer. He is also the first one to proofread my manuscript.

Table of Contents

List of Tables

List of Figures

Glossary

AdaBoost Adaptive Boosting xii, 6, 7

CNT-HOG contour descriptor based on HOG 6

EL MSER MSER region detection method that outputs center locatioin, axes and long axes orientation of the fitted ellipses. ii, 56, 91

HOG+SVM The classification method uses SVM to classify the feature vectors of HOG. 7

LBP+AdaBoost The classification method uses to classify the feature vectors of LBP. 7

PL MSER MSER region detection method that outputs pixel-list. ii, 56, 91

VGG-16 16 layers VGG network. 20

VGG-19 19 layers VGG network. 20

Acronyms

AP Average Precision 50

AVC Animal-Vehicle Collision 1, 2, 6

BIC Bayesian Information Criterion 58

Bing BInarized Norm Gradients 26

CNN Convolutional Neural Network ii, 5, 6, 19, 24, 27, 28, 30

DBA Deep Bottleneck Architecture 22, 23

DoG Difference of Gaussian 13

DPM Deformable Part Model 19, 27, 57

DT Distance Transform 8

EM Expectation-Maximization 58

FAST Features from Accelerated Segment Test 14

FPPI False Positive Per Image 50, 82

FPS Frame Per Second 89

GLM Generalized Linear Model 20

HOG Histogram of Oriented Gradient xii, 6, 7, 17, 24, 25

ILSVRC ImageNet Large Scale Visual Recognition Challenge 19, 20, 22, 24, 34

IoM Intersection over Minimum 48

IoU Intersection over Union 40, 47, 91

LBP Local Binary Pattern xii, 7, 16, 17, 19

LoG Laplacian of Gaussian 13, 15

LRN Local Response Normalization 33, 34

mAP mean Average Precision 41, 42, 44

MLP Multi-Layer Perceptron 20

MR Miss Rate 50

MSER Maximally Stable Extremal Regions ii, 5, 14, 30, 44, 45, 52, 91

NG Norm Gradients 26

NIN Network In Network 20

NMS Non-Maximum Suppression 26, 40, 64

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning 19, 49, 50, 70

R-CNN Region-based Convolutional Neural Networks ii, 5, 27, 38, 44, 52, 91

ReLU Rectified Linear Units 19, 20, 31, 33, 34

ROI Regions of Interest 25, 26, 41, 42, 52

RPN Region Proposal Network ii, 27, 43

SIFT Scale-invariant feature transform 15, 19

SPP Spatial Pyramid Pooling 27, 41

SSD Single-Shot Detector 27

SURF Speeded-Up Robust Features 15, 19

SVM Support Vector Machine xii, 6, 19, 26, 67

VGG Visual Geometry Group 20

VOC Visual Object Classes 19, 41, 50, 70

YOLO You Only Look Once 27

Chapter 1

Introduction

With more roads covering natural areas and more vehicles down the road, Animal-Vehicle Collision (AVC) has been a significant threat to road-safety and many wildlife species. For the safety of traveling public and the conservation of wildlife, mitigation measures need to be taken to reduce the number of collisions between vehicles and wildlife animals. This thesis aims to provide a novel animal detection method based on the onboard camera. With the animal detection result, drivers can be assisted to make a wise decision. The driving of unmanned vehicles can also be better guided.

1.1 Motivations and Objectives

When you google the word "hit animal", you will get millions of results. Some are asking what to do if they hit animals, some tell you what should do, the rest are trying to tell you how to avoid hitting a large animal. Although the number of this searching result may exaggerate the issue of AVC, the accidents of hitting animals indeed attract more concern nowadays. AVCs cause significant damages to vehicles, injuries or even fatality for both drivers and animals. What's more, the increase of traffic volumes, vehicle speeds, and animal populations raise the collision risk between animals and vehicles. This risk is particularly challenging in rural regions of North-America where highway traffic mileage is increasing including roads cross through the habitat of many wildlife species.

Many official statistics demonstrate the severity of AVCs. Figure 1.2 charted the mathematical numbers of AVC in British Columbia of Canada according to the records of the Insurance Corporation of British Columbia and the British Columbia Ministry of Transportation and Infrastructure [98]. In this figure, the number of motor casualty crashes, which result in an injury or fatality, exceeds 300,000 since 2012 and reaches 400,000 in 2016. The amount of fatal crash, in which a road user died, is increasing every year and reaches 300 in 2016. The number of victims in fatal crashes exceeds 300 from 2014.

In the meantime, animals suffer significant losses. 6,100 animals are recorded as killed and the unrecorded number of animal deaths amounts to 18,300 [98]. The financial loss is also alarming. The highway clean-up costs $700,000 every year. Personal, environmental,

Figure 1.1: There are 4 to 8 large animal vehicle collisions every hour in Canada [98].

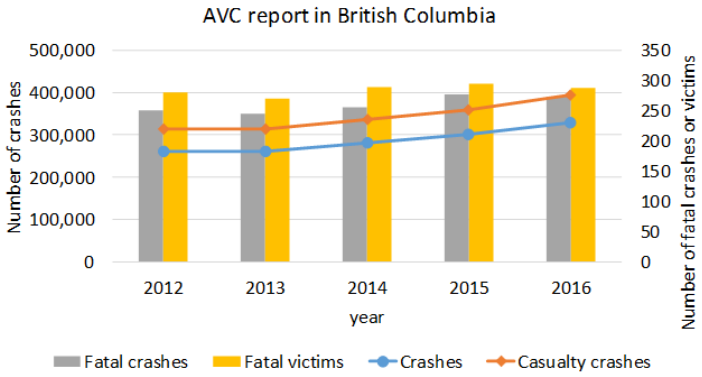


Figure 1.2: Animal vehicle collision report of British Columbia.

and financial losses are even more substantial. In the province of New Brunswick, it is estimated that the loss amounts to CA$14 million annually due to vehicle collisions with moose and deer on arterial highways [36].

In the United States, the issues go even more severe. There are approximately 200 death and 26,000 injuries of people due to AVC every year [69]. The number of wildlife victims is even more staggering. In Washington State, the study of vehicle collisions with deer and elk on state and federal highways reports at least 14,969 deer and 415 elk death between 2000 and 2004 [90]. The total annual cost associated with AVC is calculated to be $8,388,000,000 [69].

For the well-being of the population of animals and the saving of all the lost property, measures must be taken. Asking for drivers to focus on the road to avoid animals during driving is not enough. Vehicles are expected to be smart enough to localize neighboring or approaching large animals so that drivers or the intelligent vehicle system can take measures according to the position of the animals. In this thesis, we propose a novel method that detects and localizes animals in images from the onboard camera. As shown in Figure 1.3, we aim to mark the large animals out by localizing the animal and enclosing them with bounding boxes if the captured frame contains animals in the view.

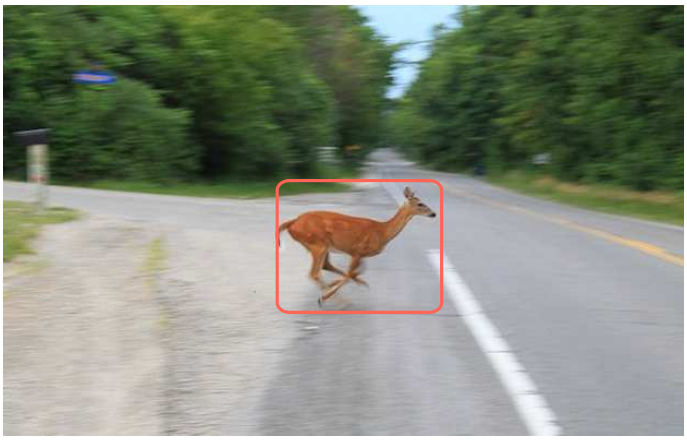


Figure 1.3: Animal detection for intelligent vehicles.

1.2 Challenges

Recognizing the existence of animal in an image is challenging. The first difficulty comes from the variant backgrounds that should be separated from foreground animals. The captured background is changing all the time along with the riding vehicle. Additionally, different segments of highways at a different time present various landscapes.

The difficulty also lies in extracting common features of various animals. On the one hand, our targets are of different species and differ greatly in size, fur color, body proportion, etc. As shown in Figure 1.4, the white horses in 1.4b are hornless, while the brown deer in 1.4b has a pair of antlers. Except for the color and antler, the on-road animals in these two figures are also different in body proportion and background. A specific detector for horses cannot be applied directly to elks.

(a) (b)

Figure 1.4: Animals in different background, color, size and ratio.

On the other hand, animals do not stay still. The poses of our targets are different when they lie, walk, run, look up or turn back, etc. Worse still, the animal can be occluded by

trees, road bulletin, vehicles and other animals. As shown in 1.5, a young elk is occluded by another young elk.

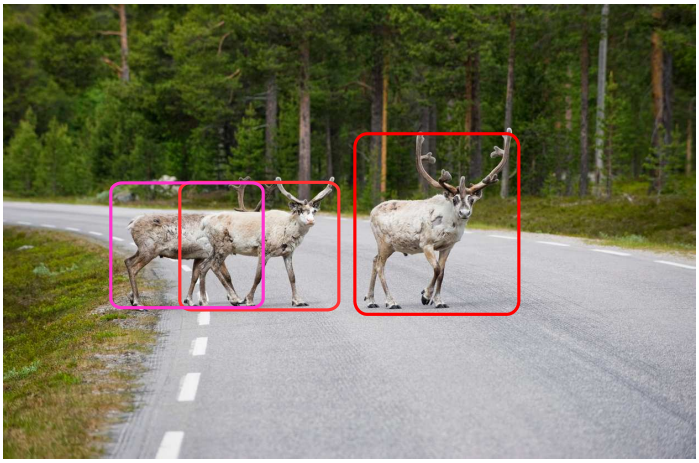


Figure 1.5: One elk is occluded by another elk.

A third difficulty sources from the processing of images captured from the onboard camera. Any imaging factor has an impact on the intensity of frames. Such factors include but are not limited to the brightness, resolution, and luminance. As a result, all the variant cases in background, animals' diversity and occlusion, and imaging factors make it difficult to craft the general features of animal for recognition and localization.

The task of animal detection for on-road vehicles is even more challenging considering that real-time results should be provided. The idea of localizing animals is far away from the requirement of being real-time. The localization of specific target is conventionally implemented based on image recognition. Namely, we firstly recognize candidate patches from the whole image and then make our detection decision by ranking the recognition result. To consider all possible position, ratio and size in an image, the conventional window-sliding method would generate thousands of patches for a considerate recognition and then detection. The costly way of this conventional localization cannot reach the expectation of our real-time application. Novel methods are in need to generate a small number of candidate patches.

1.3 Thesis Outline

This thesis is organized as follows.

Chapter 2 starts with the review of existing research works related to animal detection. We then review the state-of-the-art techniques in image classification and object detection that inspired the idea of our proposal. For image classification, we divide the techniques into three groups according to the feature involved: shape matching using shape-related

descriptors, feature crafting including extraction and description of features, and feature learning based on Convolutional Neural Network (CNN). For object detection, we firstly review the existing region generating methods and then have a look back to the object detection approaches based on CNN.

Chapter 3 studies the related basic theory of image classification, object detection and the novel region proposal method in detail. For image classification, we study the network of AlexNet and ZFNet specifically. For object detection, the thesis focuses on the family of Region-based Convolutional Neural Networks (R-CNN). The novel region detection method of Maximally Stable Extremal Regions (MSER) is introduced in the following part. We also present the measurements used for image classification and object detection, as well as the evaluation method for region proposal.

Chapter 4 proposes to employ MSER in the framework of R-CNN and fast R-CNN. We elaborate the preprocessing of the resulted MSER regions in three main operations: removing undesirable regions, clustering overlapped regions as well as modifying and enlarging regions. We then specify the following CNN network into which the regions are fed for further classification and regression. The animal dataset created is also briefly introduced.

Chapter 5 displays the experiments with multiple detailed results. We first analyze the parameters and options that may affect the detection results. The proposed detection method is then evaluate by both visualized results and multiple measurements. We also compare the result of our proposed detection method to the state-of-the-art faster R-CNN in precision, quality of detection and time-consumption.

Chapter 6 concludes with both advantages and disadvantages of the proposed detection method. Some possible enhancements for future works are discussed as well.

Chapter 2

Related Work

The methods of animal detection arise from the techniques of image classification and object detection. In this chapter, we first review the existing research on large animal detection in vehicular context. We then look back significant achievements in the field of image classification and object detection. The techniques of image classification are separated into three categories according to how the features are extracted. At last, we guide the review of related works on object detection, including region generating methods and the framework for object detection, especially those based on Convolutional Neural Network (CNN) that inspire the contribution of this thesis.

2.1 Animal Detection

There exists many methods to mitigate the AVC. Passive methods aim to warn animals away by deterrence, including ultrasonic noise such as whistles, generating high-intensity light, and road-side refraction equipments [82, 88]. The research of active methods by animal detection is still limited.

Burghardt *et al.* adopted the method of Viola-Jones human face detection successfully to detect animal faces [31]. They utilize a set of Haar-like features in AdaBoost classification and then track the detected animal face regions using a low-level feature tracker—the Kanade-Lucas-Tomasi tracking method. The integration of animal face detection and tracking achieved smooth and accurate result when testing lion faces in the real-time video. However, the animal face detection has a significant drawback that it is not a general method for animal detection, particularly for cases when animals' faces are occluded.

Ramanan *et al.* [99] built a 2D deformable model —*pictorial structure*, from animal videos and augmented the structure with a discriminative texture model. The pictorial system shows a good performance on two datasets and videos with single animal and less background.

Based on HOG, Zhou *et al.* developed a contour descriptor CNT-HOG to describe animal shapes [129] and classified the features using trained SVM classifier. The experimental

results of CNT-HOG revealed higher accuracy and efficiency than the basic HOG+SVM method. CNT-HOG is creative in utilizing infrared thermal images. However, as it is hard to extract clear contours in general images especially for cases with a complicated background, CNT-HOG cannot be easily applied in general camera images.

Zhou did systematic research on feature extraction and classification to detect on-road animals [130]. Harr-like features can be computed efficiently with the help of the integral image. HOG is good at describing edge and contour while LBP has been doing well in texture description. Through the experiments on all three features in AdaBoost classifier, he employed the LBP+AdaBoost as the first stage of the proposed double-layer animal detection system. HOG+SVM was used at the second stage. On the animal dataset, his two-stage architecture outperformed single stage methods. However, the double-layer detection system showed a limited capability at night-time and lacked the detection for postures in posterior and anterior views.

2.2 Image Classification

To recognize and then localize animals, we need to start with the overview of general approaches for image classification and object detection. The task of image recognition[1] is to identify the image with correct labels, such as animal or background in our context. The basic action for the identification is feature extraction and matching. According to the features involved, we separate the methods of image classification into three categories.

(1) Shape matching. Shape related features like edges or contours are extracted and described. Matching is then performed between the shape description and some predefined shape templates for classification.

(2) Local feature crafting. Local features such as points of interest are extracted and described by fixed-length vectors or matrices. These vectors or matrices are then entered in a trainable classifier.

(3) Feature learning. Abstract features are learned through a neural network model. The feature extractor and classifier are both contained in the neural network model.

2.2.1 Shape Matching

The shape of an object is one of the most signification visual features[2] and contains semantic information. Shapes are noticeable to a human but invisible to a computer. Besides, the geometric transformation makes the same shape display in a different manner. Thus, separable yet straightforward representation of shapes is needed for matching purpose.

[1] In some literature, image recognition is different from image classification. But in this thesis, we will take these two concepts as the same.

[2] Other visual features of the contents in images and videos include color, texture or motion, among others.

There have been numerous techniques in shape feature extraction and representation [127, 125]. This section focuses on shape description and matching methods through similarity measurements between shapes.

The Hausdorff Distance

Shapes can be represented intuitively by sets of points which consist of the contour or edges of an object. These point sets can be simply generated by some low-level feature detectors, such as a Canny edge detector. The *Hausdorff distance* is used to measure the similarity for point-to-point matching between two shapes [70]. Given two shapes A and B, where $A = \{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_p}\}$ and $B = \{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_q}\}$ are denoted as set of points, the Hausdorff distance between them can be calculated by Equation 2.1.

$$H(A, B) = \max(h(A, B), h(B, A)) \tag{2.1}$$

where

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|\mathbf{a_i} - \mathbf{b_i}\| \tag{2.2}$$

and $\|\cdot\|$ denotes the norm of the points.

From its definition, the Hausdorf distance is sensitive to even one outlier in the contour of A or B. To avoid the affection of noise, the k^{th} ranked distance is used to replace the maximal of $h(A, B)$ and $h(B, A)$ [70], as shown in Equation 2.3.

$$h^k(A, B) = k^{th}_{a \in A}\{min_{b \in B}\|\mathbf{a} - \mathbf{b}\|\} \tag{2.3}$$

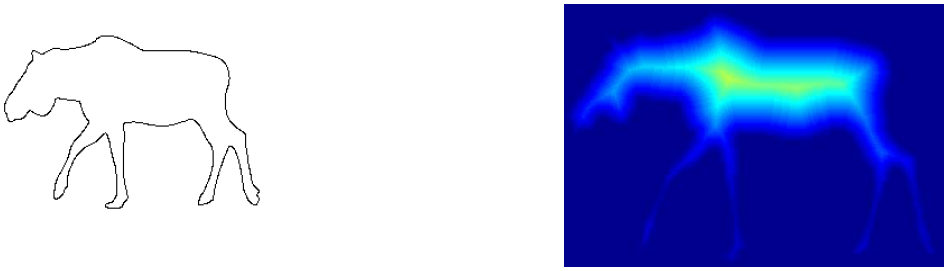
The modified version is called partial directed Hausdorff distance as given by Equation 2.4, where k_F and k_R control the fraction of the forward and reverse distances respectively.

$$H^{k_F k_R}(A, B) = \max(h^{k_F}(A, B), h^{k_R}(B, A)) \tag{2.4}$$

As the Hausdorff distance is not invariant to translation, scale and rotation, the shape matching process has to place the template shape on different positions of the image in different orientations and scales [127]. The computation in this procedure is heavy. Rucklidge proposed an efficient method to locate an affine transformation of a template in an image by applying modified versions of the Hausdorff distance [105]. Although a hierarchical rasterized search method is introduced to implement the efficient objects locating, the matching computation is still expensive [35].

The Distance Transform

The speed of shape matching based on Hausdorff distance is limited due to the complex computation and minimization. The Distance Transform (DT) is used to significantly speed up the process. The Distance Transform of an image is the process of converting a binary image to an approximate distance map [9]. In the resulted map, each pixel value represents the distance to the nearest feature pixel. DT is usually calculated by discrete



(a) (b)

Figure 2.1: The distance transform. (a) Edge of a moose. (b) The distance transform map of 2.1a

functions reflecting the minimal Euclidean distance of each pixel from the boundary pixels. As shown in Figure 2.1, 2.1a is the feature extracted by a Canny edge detector and 2.1b is the DT computed from 2.1a.

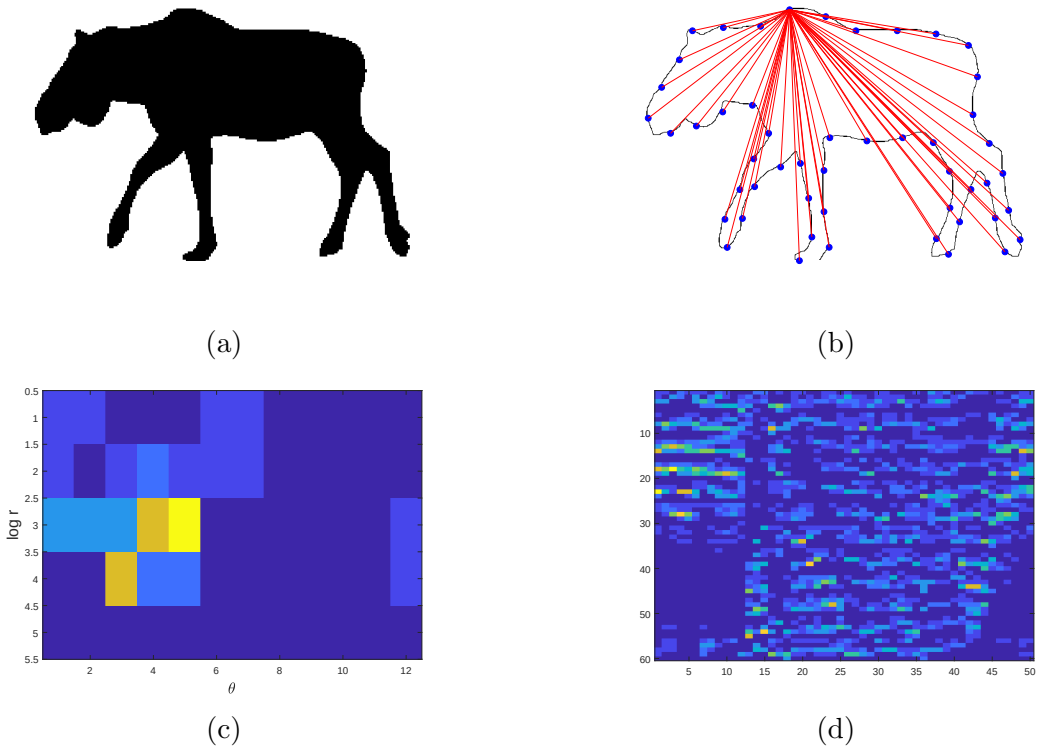
As the exact Euclidean distance leads to expensive time-consumption, the approximation to Euclidean distance is usually used. Chamfer distance has been proposed to provide effective approximation and efficient matching [35]. It reduces a translation invariant matching from $O(N^2)$ to $O(N)$, where N is the number of image points.

Gavrila [55] designed chamfer matching system using the chamfer distance. A template tree is generated offline by clustering methods so that the on-line matching traverses coarse-to-fine through the template tree and over the transformation parameters. The chamfer system is applied to realize a real-time pedestrian detection.

The Shape Context

Belongie *et al.* [8] proposed *shape context* to improve the similarity measurement of traditional Hausdorff distance. The shape context is calculated for n points picked from the shape contours. For each point, the Euclidean distance r and the angle θ to all the other $n-1$ points are computed. We then normalize the distance by the median distance, and normalize the angle in relative to the positive x-axis. Counting the number of points inside each log-polar bin, we obtain the histogram—the shape context, for the specific point. The computation process of shape context is shown in Figure 2.2. Each row of the context map in Figure 2.2d is the flattened histogram of each point context. Additionally, the matching step needs to compute the cost matrix to minimize the total cost and model the transformation. The summation of shape context distance, appearance cost and transformation cost should be minimized to find the best match.

The shape context descriptor is invariant to translation and scale due to the normalization by mean distance of n^2 point pairs. It can also be made invariant to rotation by the local tangent orientation. Furthermore, it is tolerant to small affine distortion in log-polar



(a) (b) (c) (d)

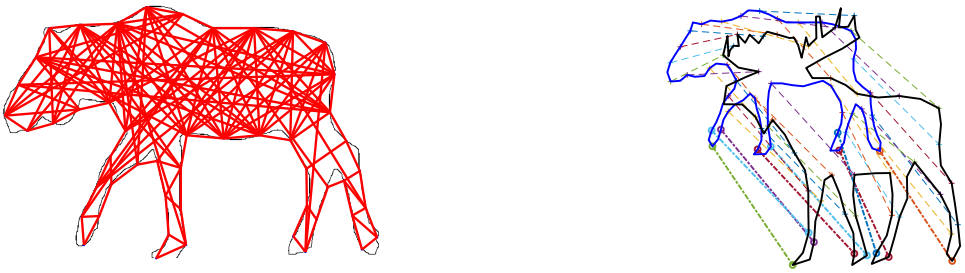
Figure 2.2: Shape context description. (a) Moose shape. (b) An edge point and all vectors started from this point. (c) Log-polar histogram of the vectors in 2.2b, the histogram is the context of the point. (d) The context map of the whole edge.

domain where the spatial blur is proportional to r. Shape context solves all problems of invariance except for articulation.

To design shape descriptor insensitive to articulation, Ling *et al.* proposed *inner-distance* defined as the length of the shortest path within the shape boundary [77]. The relative change of inner-distance during articulation is small which explains the articulation insensitivity. As shown in Figure 2.3a, the landmark points for the moose edge are connected by inner paths, which constitute the inner-distance. Compared to the connections in Figure 2.2b, paths beyond the landmark points are pruned.

The shape context is extended by replacing Euclidean distance and relative orientation with the inner-distance and *inner-angle* respectively. The inner-distance shape context descriptor describes shapes more informatively. As shown in Figure 2.3b, the dotted lines are connecting the corresponding points in the two contours of different size. For keypoints in legs with non-rigid deformation, two shapes can be connected accordingly by inner-distance shape context.

Inner-distance is robust to disturbances along boundaries, thus applicable for complicated shapes with part structures. The new descriptor of inner-distance shape context can be used for shape matching and comparison together with dynamic programming [78]. However, as the shape boundary is assumed to be known before the description, inner-distance is limited to images that can be segmented easily. Furthermore, inner-distance



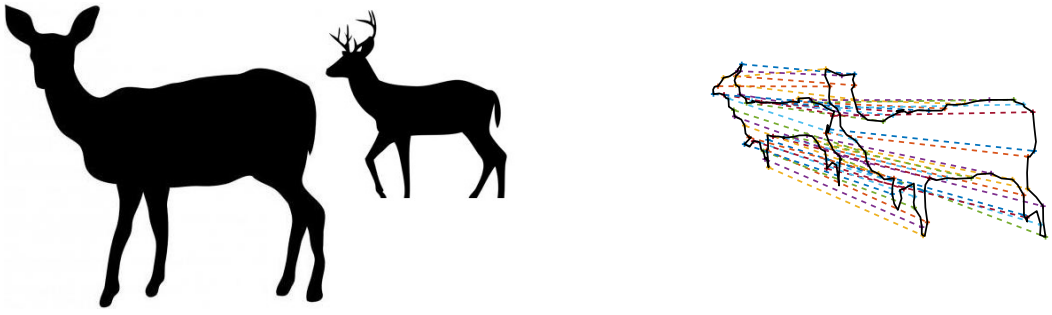
(a) (b)

Figure 2.3: (a)Inner-distance for edge points.(b) Points-matching through inner-distance shape-context.

relies on the inner-connection of shapes, any change in the inner structure, such as occlusions, may cause the description to change. Additionally, the computation cost still cannot satisfy the real-time application.

Summary

After years of development, the representation of shape features is capable to describe complicated shapes and even become invariant to affinity and articulation. However, some disadvantages of shape matching still exist. In real applications, it is difficult to extract clear shapes. In terms of animals, they are in different backgrounds and maybe occluded by other objects. The extraction of clear contours is extremely challenging. As shown in Figure 2.4, the shape matching is performed to two deer silhouettes which are of different size and pose. Edges or contours can be easily obtained from black-white images. However, as shown in Figure 2.4b, contours of the two deer are truncated with legs or tails missing, although the matching result seems acceptable by connecting corresponding points. Images of grayscale or RGB color suffer much more distortion in extracting contours.



(a) (b)

Figure 2.4: Inner-distance shape context matching. (a) Silhouette of two deers. (b) The matching result of the longest contour extracted from (a).

Another issue is that shape description is not informative enough. Different classes

cannot be separated by matching only shapes, especially when they have similar shapes in various perspectives. Other features like color, intensity, and some other local features of the shape region should be used in combination with shape features.

2.2.2 Local Feature Extraction and Description

Conventional image classification techniques can be split into two parts: feature extraction and the following classifier, as shown in Figure 2.5. The feature extractor gathers relevant information from the input image, while the classifier categorizes the resulted feature vectors into specific classes. Both parts have been studied widely for decades. In this section, we review the important techniques in extracting and describing local features.

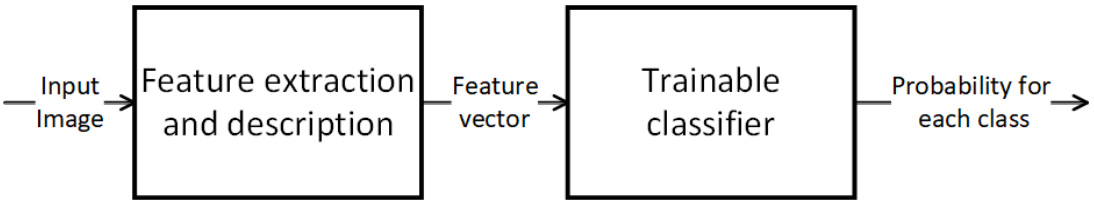
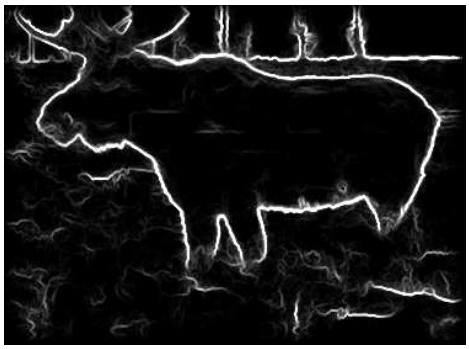


Figure 2.5: Conventional image classification model.

Local Feature Extraction

Specific image patterns or distinct structures, including edges, corners, and regions, are all local features. To recognize an image by local features under different viewing conditions, we should extract repeatable and informative local features. An ideal local feature is informative enough to separate from different classes and also is robust enough to not separate from the same class [115].

(a)



(b)

Figure 2.6: Edge detector.(a) Edges extracted by Canny detector. (b) Boundaries extracted by Berkeley detector.

Edges are the most fundamental and intuitive local features, which represent abrupt changes in some low-level image features such as brightness, color, and direction. In practice, we extract edges as sets of points with strong gradient magnitude. Canny edge detector [33] and its numerous variations [42, 124] are still state-of-the-art algorithm in edge detection. As shown in 2.6a, the edges of the moose, bushes, and trees in the background can be extracted by the Canny edge detector.

The result of edge detection provides redundant minor contours of small objects. We can address the major object boundary using a boundary detector. Strictly speaking, a boundary extends the edge detector but is different from an edge. A contour in the image plane is defined to represent a change between pixels, especially the change in pixel ownership from one object or surface to another. Berkeley boundary detector [83] eliminates noise and clutter effectively from the edge image and maintains mainly object boundaries. As shown in 2.6b, the primary contour of the moose is extracted more clearly by the boundary detector. Nevertheless, the Berkeley boundary detector is relatively time-consuming. For real-time applications, boundary images must be provided beforehand.

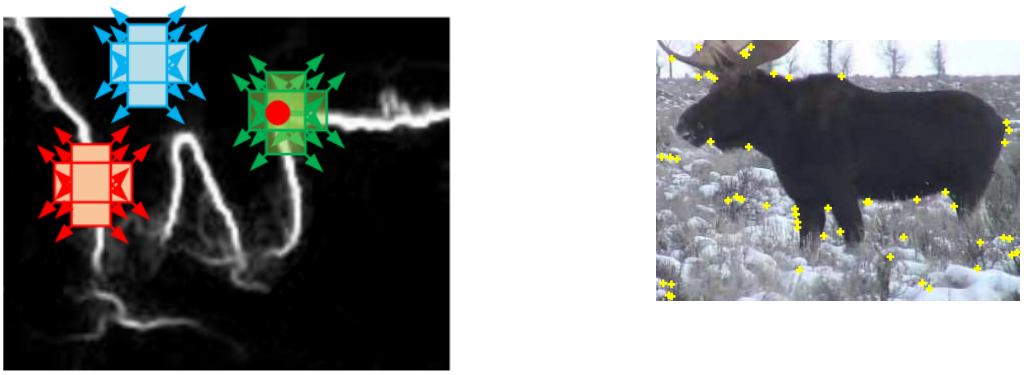
Corners are intersections of two edges. To detect both edges and corners based on intensity changing, Harris uses an auto-correlation function to measure the difference between a small window and windows shifted in any direction [62]. A matrix related to the auto-correlation function is employed in Harris detector to capture the structure of the neighborhood, as given in Equation 2.5, where (x_k, y_k) denotes the points in window W centered on (x, y) and I the image function.

$$A(x,y) = \begin{bmatrix} \sum_W (I_x(x_k,y_k))^2 & \sum_W I_x(x_k,y_k) I_y(x_k,y_k) \\ \sum_W I_x(x_k,y_k) I_y(x_k,y_k) & \sum_W (I_y(x_k,y_k))^2 \end{bmatrix} \tag{2.5}$$

As shown in Figure 2.7a, significant changes can be found in all directions in the neighborhood of the green window, which is the case when matrix $A(x, y)$ is of rank two, and both of its eigenvalues are large. In the neighboring area of the red window, no change can be detected along the edge, which indicates that matrix $A(x, y)$ is of rank one. No changes are detected in the flat region of by the blue window, which is the case when matrix $A(x, y)$ is of rank zero.

As shown in Figure 2.7b, the Harris corner points marked with yellow are keypoints which denote significant intensity changing within the local area. Harris corner is repetitive and informative [108], but it is not applicable in recognition or matching for images of different scales as the Harris corner detector only examines images at a single scale.

To gain scale-invariance, Lindeberg proposed to detect junctions by searching for local maximal value in scale space using the Laplacian of Gaussian (LoG) operators [80]. He built the scale-space representation of the image by successive smoothing the high-resolution image with Gaussian-based kernels of different size. Using Difference of Gaussian (DoG) to approximate LoG with less computation cost, Lowe obtained efficient scale-invariant corner detection by convolving the image with a DoG kernel at multiple scales [81]. To gain affine-invariance, Mikolajczyk *et al.* proposed *Harris-Affine* detector [86] based on the *Harris-Laplace* detector, which combines Harris corner detection and local maxima of LoG responses over scales [85].



(a) (b)

Figure 2.7: Corner detector.(a) Harris detection. (b) The resulted Harris corners.

In spite of the improvement in invariance, corner detectors of Harris family are computationally intensive for real-time applications. The use of machine learning breaks new ground for the research of local features and feature detectors. Rosten *et al.* proposed Features from Accelerated Segment Test (FAST)—a high-speed corner detection based on machine learning [103]. The enhanced version FAST-ER improves the repeatability. Although FAST detector is fast enough, it is not robust to noise, and it is effort-consuming to calibrate the parameters.

Another important local feature is the blob—the region where some properties are constant or approximately constant. Blob detectors detect areas in an image which are too smooth to be detected by corner detectors. Matas *et al.* proposed Maximally Stable Extremal Regions (MSER) [84] to detect blobs as connected components of a series of appropriately thresholded images. The set of *extremal regions* is decent in that it is closed under continuous transformation of image coordinates and monotonic transformation of image intensities. As shown in Figure 2.8, different regions are extracted with marking colors. MSER algorithm can also be adapted to color images. Instead of thresholding the intensity function, color version MSER uses agglomerative clustering based on color gradients [53].

MSER region is sensitive to image blur [87] which undermines the stability criterion. To be robust to blur and scale changes, MSER algorithm can be improved by detecting MSERs in multiple resolutions [54]. The MSER method works best for structured images which can be segmented well. The reason is that an ideally structured image has uniform regions that can be separated by strong intensity changes.

MSER is efficient as its complexity is near linear. Thus in practice, it is a fast detection algorithm for an affine-invariant stable subset of extremal regions. It has been mainly used to recognize or match specific objects. MSER is used in this thesis for candidate region generation and will be elaborated in later sections.

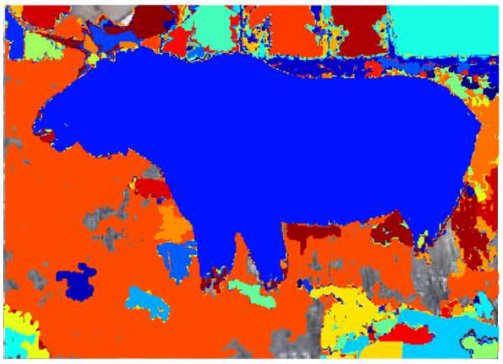


Figure 2.8: MSER region detection.

Local Feature Description

A feature descriptor seeks to represent the detected local features, such as edges, keypoints, and regions. It encodes the interesting information to a series of numbers to differentiate one feature from another. The description is expected to be invariant under image transformation.

Some feature descriptors came up with their corresponding feature extractors. The Scale-invariant feature transform (SIFT) keypoints detector is accompanied by a SIFT descriptor to describe the keypoints [80]. SIFT features are scale-invariant as interesting points are detected in various scales of the image. Rotation-invariance can then be gained by assigning an orientation to each keypoint that takes higher contrast than a threshold. As shown in Figure 2.9a, four sample SIFT keypoints are detected using different scales and orientations. The length of radius for the circles reflects the impact of scale, and the direction of the arrow reflects the impact of orientation.

SIFT descriptor starts by dividing a 16×16 neighborhood around the keypoint into 16 sub-blocks of size 4×4 and then generates an 8-bin orientation histogram for each sub-block. The descriptor outputs a SIFT vector of length 128 as it contains $16 * 8 = 128$ bin values. As shown in Figure 2.9b, each SIFT feature is composed of 16 sub-blocks in which specified orientation histogram is displayed[3]. In terms of histogram magnitude, the sub-blocks in two green bocks are stronger those in two red ones.

As SIFT features and corresponding description are invariant to scale and orientation and are also robust to changes in illumination, noise, and viewpoint, The drawback of SIFT is its comparatively low processing speed. A faster version—Speeded-Up Robust Features (SURF) is raised [7]. SURF uses box filter instead of LoG to calculate convolution and uses *Wavelet* responses in horizontal and vertical direction for orientation assignment and feature description. Both convolution and Wavelet response can be implemented easily

[3]The feature blocks are magnified for better visualization.

(a) (b)

Figure 2.9: SIFT feature and descriptor.(a) The keypoints marked with yellow circle. (b) SIFT description marked with 4×4 blocks.

using integral images at any scale. SURF can be three times faster than SIFT with comparable performance [72]. SURF is good at handling images with blurring and scaling, but not robust to the rotation and illumination changes. SIFT and SURF have been widely used in keypoints extraction, object recognition, and video tracking.

Local Binary Pattern (LBP) denotes a powerful feature extractor and descriptor for local texture information [93]. The LBP of the center pixel is calculated by the neighborhood of size r in a grayscale image. As shown in 2.10, the eight neighboring points in the 3×3 neighborhood are used for the calculation. Intensity values of neighboring points are first binarized to 1 or 0 thresholded by the center pixel. The eight binary values are then read from a predefined starting point in clockwise or counter-clockwise. In this example, the binary sequence is formed clockwise using the underlined point as the least significant digit. The sequence is then translated to a decimal value which represents the LBP value for the center pixel. The LBP feature map can then be obtained by calculating all pixels in the image, as shown in Figure 2.11b.

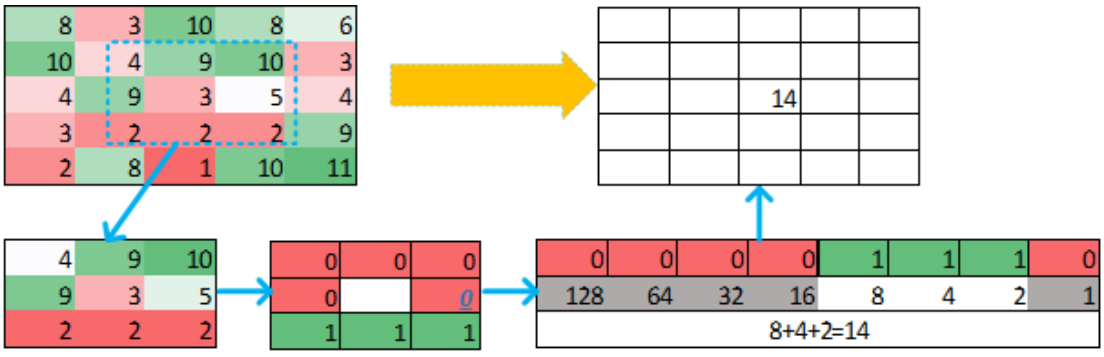
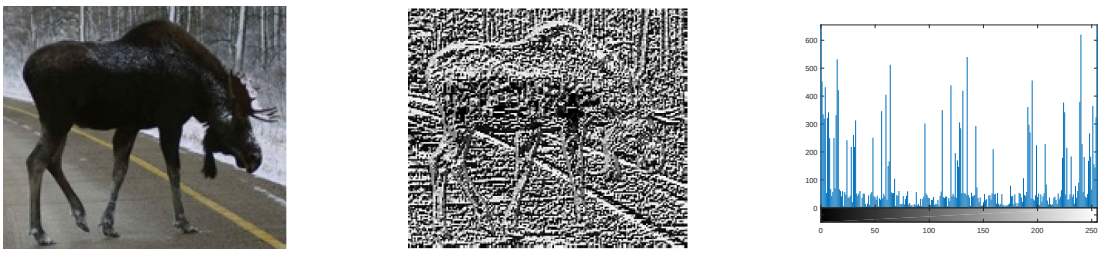


Figure 2.10: LBP description.

Except for translating the binary sequence to a decimal value, we can also achieve a fixed-length feature vector for classification or matching. Through the modification of the

radius of the neighborhood circle and the number of neighboring points, we can obtain fixed-length LBP feature vector. The LBP feature vector for the whole image can then be achieved by partitioning the image into N non-overlapping blocks and representing each block by fixed-length histogram vector $1 \times B$. The overall LBP length is $N \times B$.



(a) (b) (c)

Figure 2.11: LBP feature. (a) Original Image. (b) LBP feature map. (c) LBP histogram.

One important extension to the original LBP is *uniform pattern*. The uniform pattern reduces the length of the feature vector and benefits from rotation-invariance [6]. LBP feature is popular in face recognition and texture analysis.

Among the fruitful works of literature, Histogram of Oriented Gradient (HOG) descriptor attracts the most attention. Proposed for human detection [40], HOG represents the distribution of intensity gradients and edge directions. After obtaining the gradient image with magnitude and orientation, we then group the gradient image into cells of the same size. 6×6 is often used as predefined cell size. We divide all 360° orientations of the gradients for each cell into fixed bins to build the histogram. As shown in Figure 2.12, the orientation is separated into nine bins (each bin is 20°). The histogram of each cell is a feature vector of length 9 with each value representing the weighted sum of gradients in each bin.

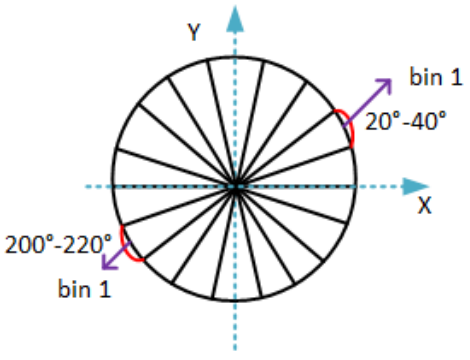


Figure 2.12: HOG bins.

The magnitudes of pixels in gradient image are deeply affected by illumination, shadow or edge of objects. The influence of these factors can be reduced by normalizing the histogram of each cell. By grouping neighboring cells into blocks, as shown in Figure 2.13 where each block is formed by 2×2 cells and overlapped with its neighboring block, we

can concatenate the feature vectors of length 9 from each member cell to produce the HOG vector of length 4×9 for the block. The concatenation of HOG vectors of all blocks is the HOG vector of the full image.

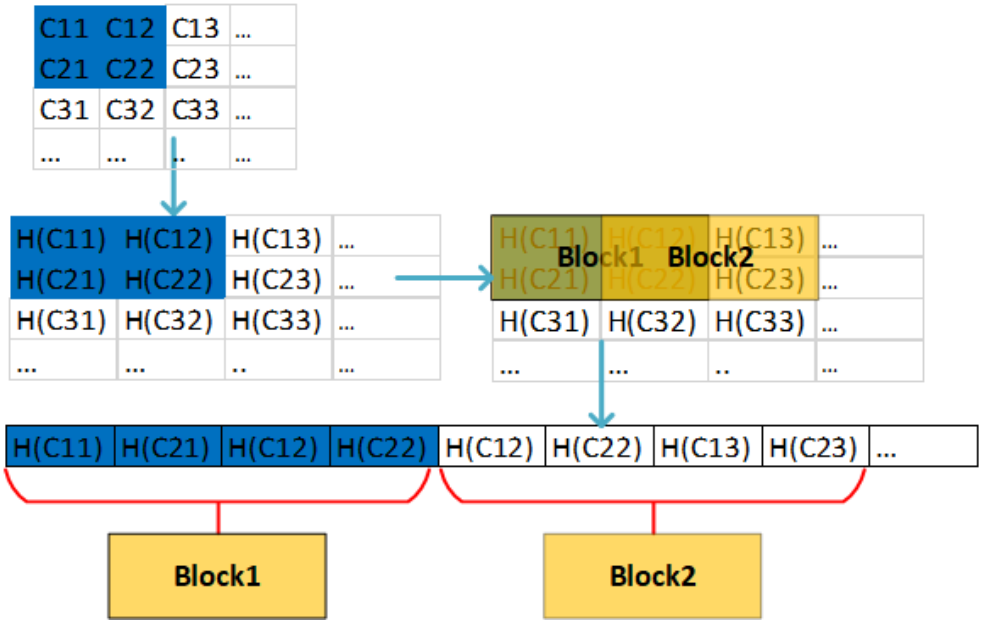


Figure 2.13: HOG description.

The HOG feature map of the moose in image in Figure 2.11a is visualized in Figure 2.14. We can still recognize the outline of the moose in Figure 2.14a. The oriented gradient histogram is displayed in a clear way in Figure 2.14b, which focuses on the head part of the moose.

(a) (b)

Figure 2.14: HOG feature map.

HOG descriptor is invariant to geometric and photometric transformations, thus is particularly suited for human detection regardless of humans' movement as long as they are roughly upright. But HOG is not invariant to orientation. Also, HOG descriptor is

costly in the computation on a dense grid of uniformly spaced cells and in the overlapping local contrast normalization.

Using single rigid template as model, HOG descriptor is not invariant to deformation. Modeling the parts of the objects, Felzenszwalb and Girshick proposed Deformable Part Model (DPM) to gain deformation invariance [51]. DPM consists three types of descriptors. Besides the basic description of the whole object using HOG, DPM describes a number of parts using HOG. With positional variation in the parts, a third descriptor is used to penalize the deformation of the parts with respect to the optimum positions.

DPM benefits from the ability to handle parts deformation. The method of DPM detection won the PASCAL VOC 2007 challenge. However, HOG description for parts needs twice the image resolution of that used for the basic HOG description. The computation of DPM is significantly expensive for practical applications.

Summary

In this section, we reviewed the main progress in feature crafting including feature detection and feature description. Features and corresponding descriptions can be invariant to scale, rotation and deformation. In practice, we extract local features first and then describe the keypoints using SIFT, SURF or LBP. The resulted feature vectors or matrices are then fed into trainable classifiers for classification, such as k-Nearest Neighbor (kNN), Support Vector Machine (SVM), Random Forest. The classifier part is merged into the feature learning technique as explained in next section.

2.2.3 Feature Learning

The conventional image classification needs to classify the features which are extracted and described in crafting manner. The state-of-the-art techniques learn features and train classifiers through the neural network. This section focuses on image classification based on Convolutional Neural Network (CNN).

As early as 1988, LeCun had applied CNN in pattern recognition and proved that CNN played the role of hand-crafted feature extractors. The first CNN structure *LeNet* was proposed in 1994 [74]. The LeNet architecture contains five layers, as shown in Figure 2.15. Spatial features are extracted through convolutions. To reduce parameters, LeNet uses subsampling to get the spatial average of maps from last layers result. LeNet introduces non-linearity using activation functions, such as *tanh* or *sigmoids*. As a whole, the multi-layer neural network forms a classifier. LeNet did well in hand-written digit recognition. However, the expensive computation cost blocked the development of CNN at the time with low CPUs and no GPU.

Until the 2010s, CNN was back in fashion again due to the rise of the computing power from both CPU and GPU. In 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), the winning approach obtained higher image classification accuracy through *AlexNet* [73]. Besides stacking more convolutional layers, AlexNet uses Rectified Linear

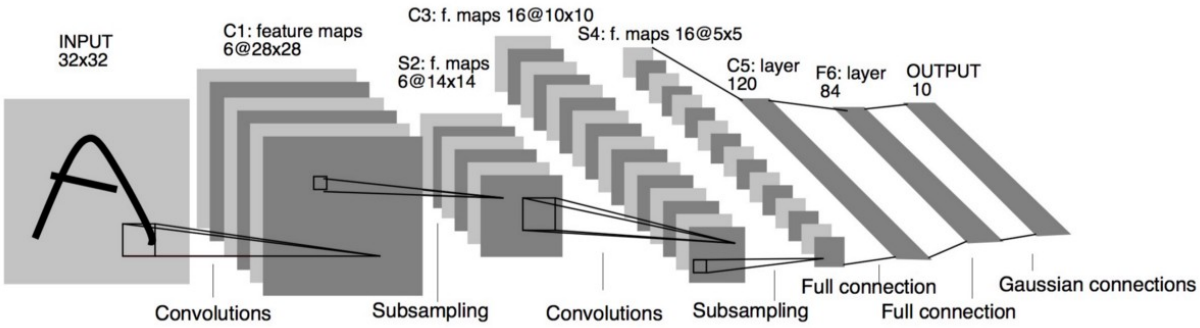


Figure 2.15: LeNet architecture [74].

Units (ReLU) instead of *tanh* for non-linearities. Furthermore, overlapping max pooling is employed to avoid the averaging effect of average pooling (or subsampling in LeNet). AlexNet uses dropout technique to selectively ignore single neurons during training to prevent the network from *overfitting*[4]. Also, the training dataset is augmented by transformations to reduce the overfitting further.

Zeiler and Fergus proposed ZFNet by improving AlexNet and won ILSVRC 2013 [126]. ZFNet modified several hyperparameters including the size of the middle convolutional layers, and the stride and filter size on the first layer. Smaller filters introduce a finer resolution on the images. ZFNet achieved a top-5 error rate of 14.8%.

The research group Visual Geometry Group (VGG) in the University of Oxford designed VGGNet and won the second place of ILSVRC 2014. VGGNet shows that the depth of CNN plays an important role [110]. A significant improvement in image classification accuracy can be achieved by increasing the depth from 8 to 16, or from 16 to 19. The 16-layer network VGG-16 and deeper 19-layer network VGG-19 are shown in Figure 2.16 and 2.17 respectively. VGGNet performs 3×3 convolutions and 2×2 pooling in all convolutional layers to represent complex features but using fewer parameters. Despite the reduction, VGGNet still needs to maintain a large number of parameters making it difficult to train.

In 2014, a deep network structure Network In Network (NIN) was proposed as an improvement to the traditional CNN [76]. The traditional convolution filter in CNN is a Generalized Linear Model (GLM), as shown in 2.18a. The NIN architecture maps the local patch to more abstract features using a nonlinear function approximation—the spatial Multi-Layer Perceptron (MLP), as shown in 2.18b. The overall structure of the NIN is the stacking of multiple MLP convolutional layers. Using ten times fewer parameters than AlexNet, NIN obtained better performance.

GoogLeNet was proposed based on NIN and won the ILSVRC 2014 [111]. GoogLeNet uses an efficient *Inception Module* to reduce the number of parameters. The naive Inception Module, as shown in Figure 2.19a applies parallel filter operations on the input from the previous layer so that multiple receptive field sizes can be achieved from the convolution of

[4]Overfitting means the network model may learn the details of the training set so well that it cannot be generalized to test images.

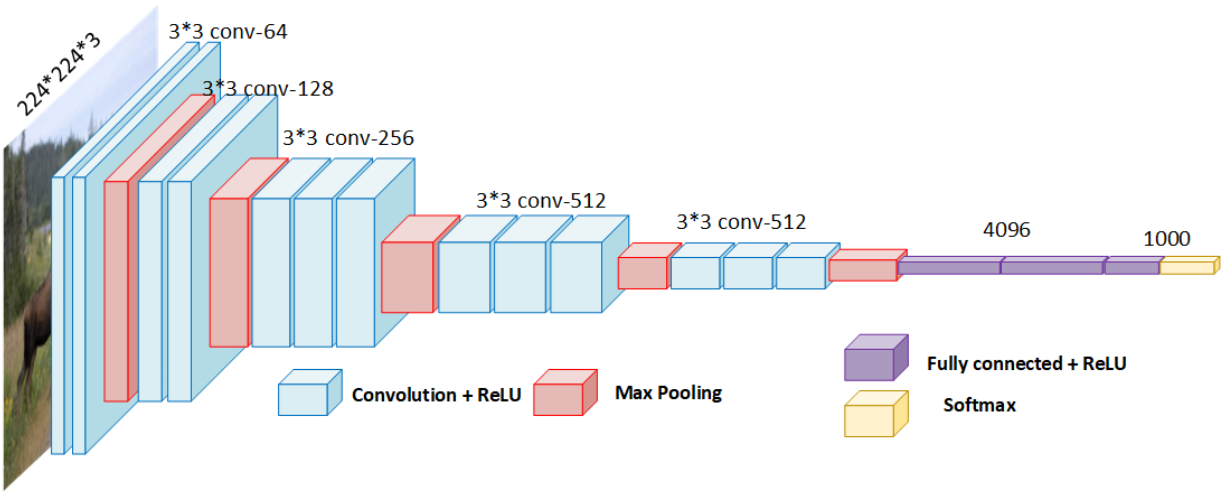


Figure 2.16: VGG-16 architecture.

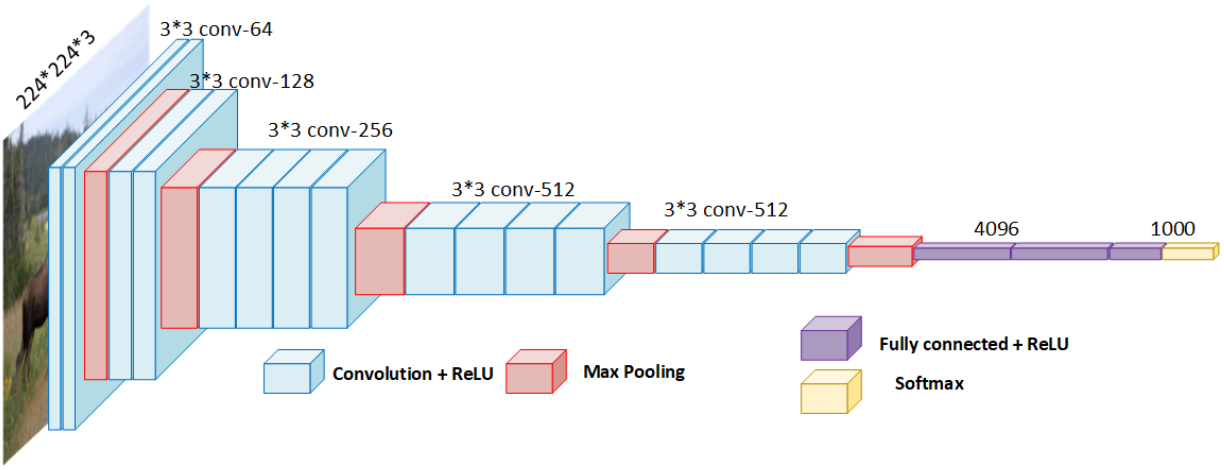
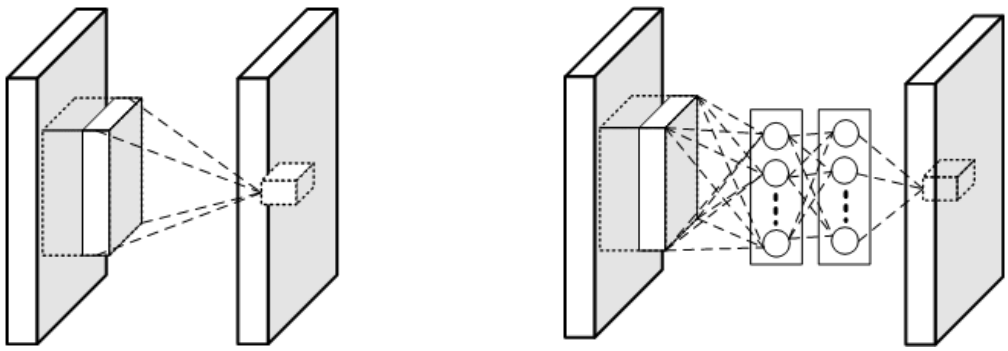


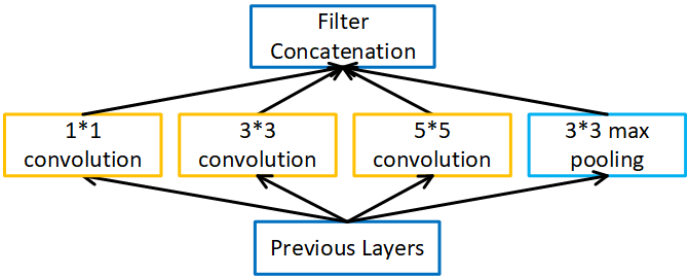
Figure 2.17: VGG-19 architecture.



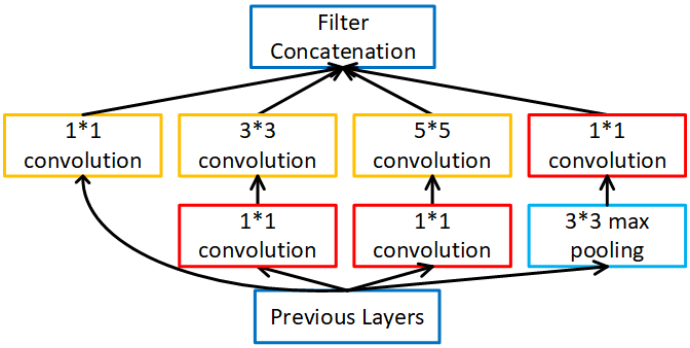
(a) (b)

Figure 2.18: Comparison of linear convolution layer and MLP convolutional layer. (a) Linear Convolutional Layer. (b) MLP Convolutional Layer.

size 1×1, 3×3 and 5×5. The naive structure has a huge computational complexity. To reduce the computation cost, GoogLeNet uses "bottleneck" layers with 1×1 convolutions to reduce feature depth as shown in 2.19b. The 1×1 convolution proposed in NIN contributes to reducing the dimension before the expensive parallel blocks of 1×1, 3×3 and 5×5. Furthermore, the 1×1 convolutions make it possible to increase the depth and width of the network. Additionally, average pooling layers plus softmax classifier are utilized to replace fully connected layers at the top of the convolutional layers so that a large number of parameters can be eliminated.



(a)

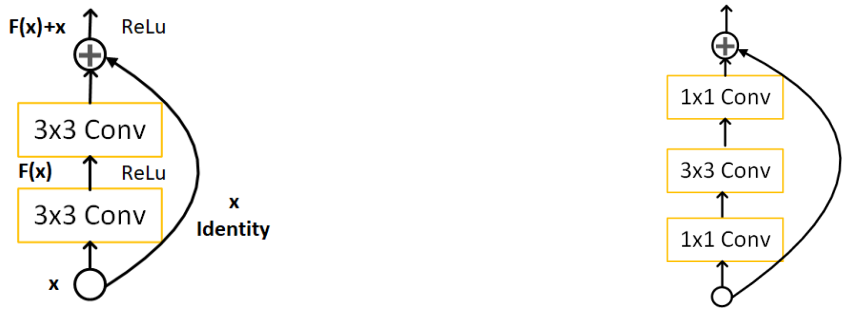


(b)

Figure 2.19: Inception Modules. (a) Naive Inception module. (b) Inception Module in GoogLeNet

With more and more layers stacked, deeper CNN does not lead to better performance but accuracy degradation. It sources from that deeper models become difficult to optimize and train. K. He proposed *Residual Network (ResNet)* to solve the problem by stacking residual blocks, which mapped the network layers to residual instead of the desired underlying input [64]. ResNet was the winner of ILSVRC 2015. As shown in Figure 2.20a, every residual block has two 3×3 convolution layers. Deep Bottleneck Architecture (DBA) can be used to generate deeper networks. As shown in Figure 2.20b, the residual block is transformed into a bottleneck architecture with three consecutive convolutional layers of size 1×1, 3×3 and 1×1 separately.

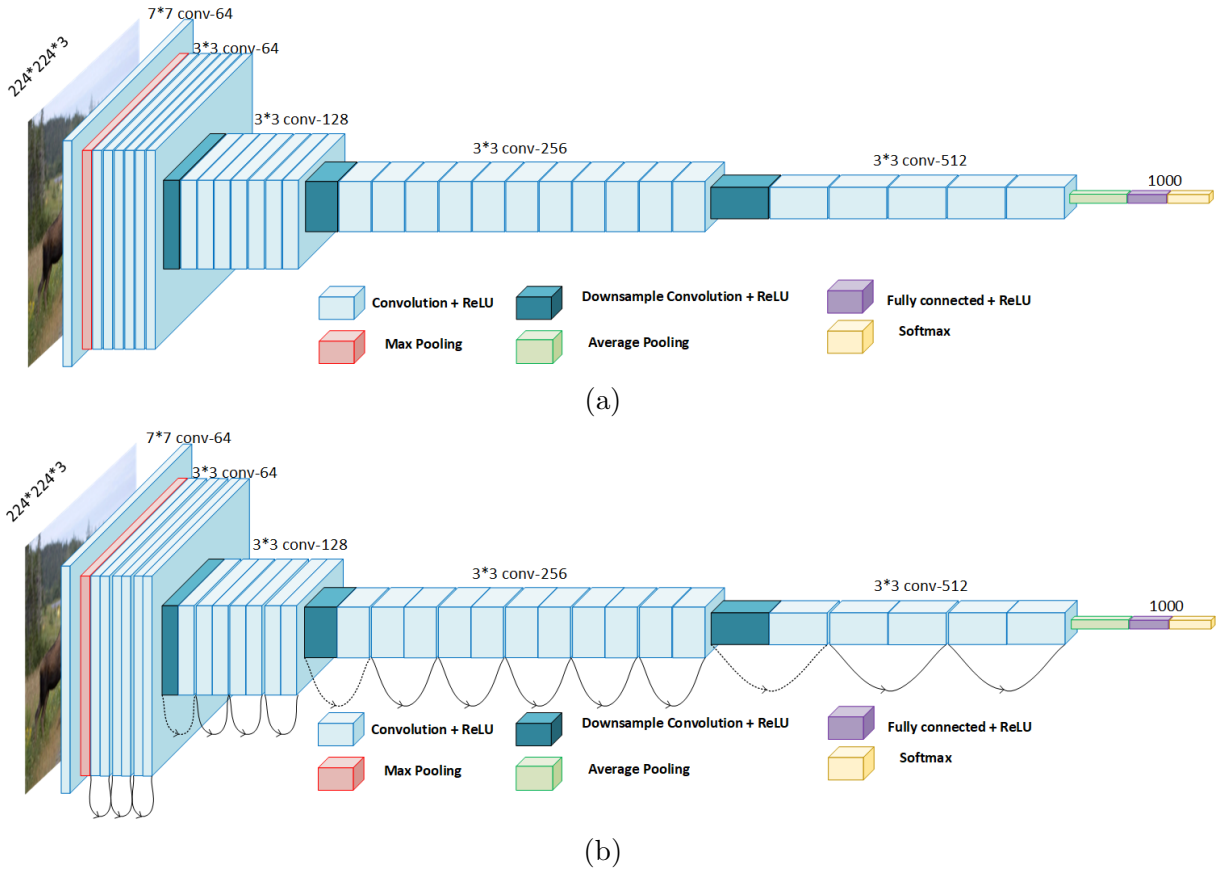
As shown in Figure 2.21b, a full ResNet with 34 layers groups every two neighboring layers into one residual block, while the plain version of 34-layer network in Figure 2.21a stacks the convolutional layers directly. Except for the residual blocks, the ResNet architecture has an additional convolutional layer at the beginning and a global average pooling



(a) (b)

Figure 2.20: ResNet Modules.(a) Residual Blocks. (b) Bottleneck Residual Blocks used in deeper models.

layer in the end. To reduce the number of parameters and thus improve efficiency, K. He implemented ResNet with DBA. As shown in Figure 2.22, the practical ResNet is built by 50 layers with bottleneck architecture, known as ResNet-50. Using similar structure, ResNet allows for more than 100 layers stacked and avoid accuracy degradation.



(a)

(b)

Figure 2.21: ResNet structure.(a) 34-layer plain network. (b) 34-layer residual network.

Deep neural networks have led to a series of breakthroughs for image classification.

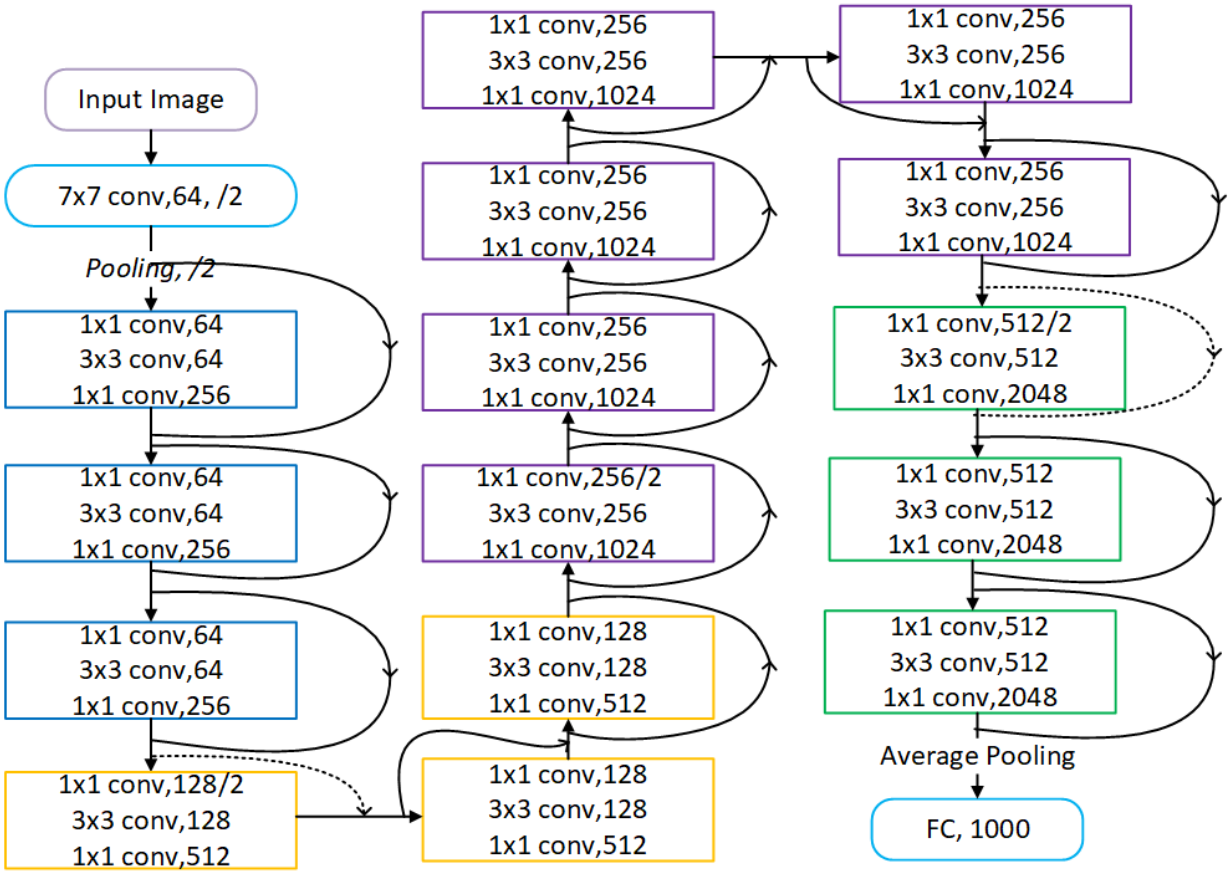


Figure 2.22: ResNet-50 architecture.

Feature extracted through CNN are more discriminative than low-level image features like HOG. The depth of network reflects the complexity of the features. As shown in Table 2.1, the top-5 error decreases along the rising of network depth. The parameters and techniques involved with the network structure are also listed in the table, such as kernel size, normalization skills, and the dropout techniques. As the winner of ILSVRC 2015, ResNet reduces the top-5 error to 3.6%, which is better than human performance. The drawback of CNN features, however, is the heavy computation. High-quality GPU is needed to for deeper models and faster implementation.

2.2.4 Summary

In this section, we reviewed the development of techniques on image classification in three categories. Shape matching is an intuitive method with expensive effort on shape description such as shape context and inner-distance shape context. Local feature crafting which combines feature detector and feature descriptor has dominated for decades in both image classification and object detection. Currently, the method of deep learning based on CNN has brought new breakthrough and becomes a research focus. This thesis proposes a novel animal detection system based on the convolutional network.

Table 2.1: Comparison of classical CNN networks.

Model	AlexNet	ZFNet	VGGNet	GoogLeNet	ResNet
Year	2012	2013	2014	2014	2015
No. of layer	8	8	19	22	152
Top-5 error	16.4%	11.7%	7.3%	6.7%	3.57%
Data Augmentation	+	+	+	+	+
Inception(NIN)	-	-	-	+	-
No. of Convolution layer	5	5	16	21	151
Kernel size	11,5,3	7,5,3	3	7,1,3,5	7,1,3,5
FC layer No.	3	3	3	1	1
Dropout	+	+	+	+	+
Local Response Normalization	+	+	-	+	-
Batch Normalization	-	-	-	-	+

It is noteworthy that the techniques among these three categories are interlinked. In the method of shape matching, the shape related features could also be considered as local features and the associated matching method can then be regarded as a classification method. For local feature crafting, DPMs are effective graphical models for visual recognition and can also be formulated as an equivalent CNN [59]. The formulation from DPM to CNN provides a synthesis view between feature crafting and feature learning.

2.3 Object Detection

While the task of image classification is to assign the correct class label to the whole image, the task of object detection is to localize the right object with an appropriate bounding box in an accurate position. Based on the idea of "recognition using regions" [61], detection approaches are mainly developed by generating numerous Regions of Interest (ROI) and then classifying these areas. This section reviews the techniques of generating object locations and the application of CNN in object detection.

2.3.1 Object Locations Generation

The conventional method to generate candidate regions is through the exhaustive search which slides a window of fixed size all over the image. By scaling down the original image with a series of factors, we can obtain the scale-space pyramid as shown in Figure 2.23. Sliding the window of fixed-size on each image in the pyramid, we can inspect windows at all scales and locations. The human detector of HOG is performed in a window-sliding manner [51].

The scale-space image pyramid sets the stage for inspecting candidate regions of different size but is still deficient in dealing with rotation variance. In the example of the human detector, people with an upright position can be detected while people lying, falling

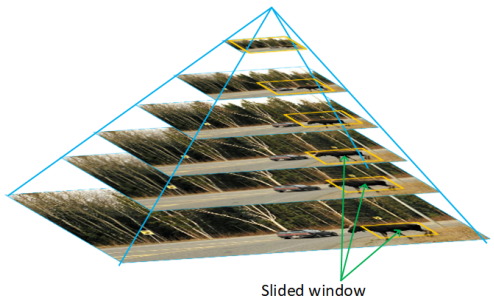


Figure 2.23: Scale-space image pyramid for window sliding.

or with other non-upright positions will be missed. Most importantly, the approach of window sliding with the pyramid generates too many regions of interest [67]. The great number of regions to be processed by the following image recognition algorithm cost high in computation.

The methods of *Region Proposal* have been widely used to generate a small number of Regions of Interest (ROI). There exists a great number of region proposal methods can can be divided into two categories: grouping methods and window scoring methods [66]. Grouping methods generate ROIs based on segments corresponding to objects in the image. Window scoring methods score each candidate region according to objectness which denotes the possibility by which an object is contained.

Selective Search is a typical grouping method which generates region proposals by merging superpixels [116]. The image is segmented into small regions through feature engineering. Then the small regions which are similar to each other are iteratively merged into one region by a greedy algorithm. Varying the starting regions and the similarity measurement helps to augment and diversify the hierarchy of the generated regions. Selective Search generates qualified regions using different grouping criteria and multiple complementary strategies to deal with various image conditions. Regions generated by Selective Search are not restricted to the aspect-ratio or size of the object. However, the proposed regions may not contain objects at all.

The window-scoring method seeks for candidate regions according to the objectness. Alexe *et al.* estimated the objectness by combining multiple complementary cues, including the conventional multi-scale saliency, color contrast between the object and background, edge density, location and size, and the straddling of segments [4]. The number of regions can be reduced by Non-Maximum Suppression (NMS) sampling.

Cheng *et al.* proposed BInarized Norm Gradients (Bing) [34] to measure the objectness. The Norm Gradients (NG) features are robust to change of translation, scale and aspect-ratio. Resizing candidate windows to 8×8, BING approximates the NG feature of the window with binary vector and train a linear SVM classifier to learn the objectness scores. A very fast class agnostic detector can be obtained through enough approximations.

Zitnick proposed *Edge Box* to score the candidate window based on the number of contours wholly enclosed by the window [131]. Zitnick obtained the initial edge map using the fast Structured Edge detector [46]. Edge Boxes provides a good compromise between

speed and quality.

2.3.2 Object Detection through CNN

The last decades witnessed significant progress in object detection. The conventional object detection system combines the local features, such as SIFT and HOG, with sophisticated machine learning frameworks. But the performance of object detection plateaued for several years until the fast development of CNN.

Girshick *et. al* proposed Region-based Convolutional Neural Networks (R-CNN) by combining region proposals with CNN features [57]. The candidate regions are generated by Selective Search and then fed into a CNN structure for feature extraction and classification. The framework ends with a bounding box regressor which refines the position and enclosing rectangle. R-CNN brings breakthrough by much better detection accuracy than DPM.

Nevertheless, R-CNN is slow as it needs to perform a CNN forward pass for each proposed region independently without any computation sharing. The slow detection is even severe when using deeper CNN structures. Moreover, R-CNN warps the proposed regions to match the input size 224×224 of the convolutional network. The unexpected geometric deformation limits the performance improvement.

K. He proposed SPPNet to eliminate region-warping by inserting Spatial Pyramid Pooling (SPP) layer between the convolutional layers and the fully-connected layers [63]. Girshick proposed Fast R-CNN by equipping an SPP-like layer—ROI pooling layer [56]. The feature map resulted from the convolutional layers are shared by all proposed regions. ROI pooling layer extracts fixed-size features for all proposed regions from the feature map. Feature vectors are then fed into the following fully-connected layers. The shared feature map speeds up the detection.

The bottleneck of Fast R-CNN is the region proposal algorithm. R. Girshick et al. replaced proposed Region Proposal Network (RPN) to generate regions [101]. RPN works on the feature map resulted from the convolutional layers of the CNN network. RPN maps each 3×3 sliding window and its anchor boxes to their objectness scores and coordinate modification arguments. Windows scored higher are selected and entered into a Fast R-CNN for further detection. Altogether, Faster R-CNN achieves much faster detection and a state-of-the-art accuracy.

The network You Only Look Once (YOLO) performs region proposals and classification in a single shot [100]. YOLO divides the feature map into 7×7 blocks to replace the operations of region proposals and feature re-sampling. The number of regions to be processed are reduced from 2,000 in Faster R-CNN to 98. YOLO increases the detection speed significantly and can be applied in real-time detection. However, the detection accuracy of YOLO is decreased. Furthermore, the block size 7×7 is so coarse that the detector does not apply to detection for small object.

Single-Shot Detector (SSD) extends YOLO by adding a small convolution filter [79]. The convolution filter predicts category scores and bounding box offsets for a fixed set of default bounding boxes. Different from anchor boxes in RPN, the default bounding boxes

are applied to feature maps of different resolutions. SSD performs faster than YOLO and produce markedly superior detection accuracy.

2.4 Summary

In this chapter, we looked back the development of animal detection, as well as the significant techniques in object detection. Remarkable achievements have been made in the area of object detection and image classification, especially after the occurrence of CNN. The network of ResNet in image classification has outperformed human. Besides, object detection based on CNN has achieved great success. The research in region proposal method and object detection network inspire the new idea for animal detection of this thesis.

The application of animal detection in vehicles is feasible and even enhanced due to the development of Vehicular Ad-hoc Networks (VANETs), which offers the power of fast computation and safe data communication. Due to the highly dynamic topology and intermittent connectivity, VANETs have to break the challenges of different application QoS requirements, and conflicting privacy and safety issues [39]. The development of VANETs technology benefits from the research of all kinds of fields, including distribution simulation, Wireless Sensor Networks (WSNs) and Mobile Ad-hoc Networks (MANETs).

The idea of distribution algorithm [29, 12] sets the fundamental for high-performance load-balancing data distribution management [27, 26, 13, 18, 3], which manages the distribution of state updates and interaction information in large scale distributed simulations.

Ubiquitous computing which provides services anywhere, anytime in an unobtrusive, seemingly invisible way extends the vehicles computing units. The potential of WSNs in UC applications attracts increasing attentions. Efficiency of network can be improved by modified routing protocols in speeding up the relay of sensed data [20] or by scalable and dynamic routing protocols in data aggregation and collection [41, 121, 119, 123, 120]. Due to the broadcast characteristic of wireless mediums, the opportunistic routing paradigm is widely used to improve network performance in wireless network, as well as in underwater sensor networks [11, 37]. Using short distance transmissions by employing nearest neighbor nodes between neighboring clusters propagates messages in an energy-efficient way [17]. To track or match the sensing nodes, the location can then be estimated by localization algorithms with referenced data [94, 19, 41] and low energy dissipation [106]. The coverage plays an important role in quality of service and could be improved using various methods [15, 14]. For networks of small dust, a varying range of data transmissions contributes to high fault-tolerance of WSNs.

MANET consists of a collection of some independent heterogeneous mobile or stationary hosts interconnected via wireless links. A self-diagnosis MANET can be obtained by comparison approaches [48, 49]. The reputation of the mobile nodes can be evaluated to supply with robust trust services [22, 23]. The routing efficiency for MANET and data mining approaches also fuels that in VANETs [25, 28], as well as robust data dissemination for VANETs [1, 122]. For tracking and correlation purposes especially for estimating the distance of detected objects, the correct and exact location information of vehicles is

also required. Except for the equipped GPS, localization of neighboring vehicles and the cooperation with them not only provide more accurate and detailed information, but also overcome a nonline-of-the-sight condition and secure the integrity of localization services [24, 2, 1].

Data communication and application in VANETs are challenging due to the mobility of vehicles, limited hardware equipped onboard especially for the multimedia data with huge size. The synchronization scheme of video streaming plays an important role in real-time video processing [16]. The streaming efficiency can be improved by moving the demanding task to a dedicated remote server and leaving only the minor image-based tasks to the local [21]. The imagery data can also be prestreamed using PROgressive PANorama StrEaming protocol (PROPANE)[95]. Cristiano et al proposes a novel protocol VIRTUS to offer a reacitve and scalable unicast solutions [102]. The combination of algorithms in image processing with the techniques in VANETs and data mining makes the application of live-video-streaming animal detection and track feasible and promising.

Chapter 3

Basic Theory

In this thesis, we propose a novel animal detection method by combining feature crafting and CNN network. In this chapter, we first introduce the CNN structures in image classification and then present the family of R-CNN in object detection. We also introduce the Maximally Stable Extremal Regions (MSER) as a new region proposal approach.

3.1 CNN for Image Classification

The structure of regular Neural Network is constructed by fully connected layers, where all neurons are fully connected with all neurons in the previous layer. The full interlayer connection leads to a large number of weights. Besides, the neurons in each layer are learned independently without any shared connections. Therefore, the general Neural Network is hard to train yet easy to cause overfitting.

The structure of Convolutional Neural Network (CNN) improves the regular Neural Network by introducing more complex layers and adopting new techniques. Instead of connecting to all neurons in a fully-connected manner, neurons in a CNN layer are connected to a small region of the previous layer and arranged in three dimensions with width, height and depth[1].

3.1.1 Layers in Convolutional Neural Network

A CNN used in image classification transforms an input image of 3D to a vector of classification score. A sequence of layers is used to construct the structure of CNN, including convolutional layers, activation layers, pooling layers and fully connected layers.

[1]Depth of the neuron is not the depth of the network. The latter denotes the number of layers in a network.

Convolutional Layer

The convolutional layer is a trainable 3D filter of size. Assuming that the width and height are identical, we can denote the size of the filter as $F \times F \times D$ and regard the 3D filter as an array of 2D filters[2] of size $F \times F$. Similarly, we denote the size of the 3D input volume as $W \times H \times D_{in}$ and regard the 3D input as D_{in} channels of 2D spatial data. The convolutional layer works by convolving each $F \times F$ filter with each spatial input to generate a total of $D \times D_{in}$ response maps as an output volume. The size of the receptive field in the input volume represents the resolution of the CNN feature and equals to the size of 2D filter. For each specific 2D filter, the learnable parameters are shared through all spatial positions in the input volume.

The use of zero-padding to the input volume during convolution contributes to extract feature around the border. Decreasing the striding steps helps to obtain refined feature. Besides the affect to the performance of the convolution layer, zero-padding and striding also have an impact on the spatial size of the output volume. For an input volume of size $W \times H \times D_{in}$, the width W_{out} of the output volume can be calculated as $W_{out} = (W - F + 2P)/S + 1$, where S and P denote the striding steps and the size of zero-padding. The numbers of parameters for the convolutional layer can be also calculated. With symbols above, a total of $F \times F \times D_{in} \times D$ weights and D biases are needed. The increasing of filter size, amount of filters and number of layers will consequently increase the number of parameters.

Activation Layer

Activation layer introduces nonlinearity to the transmitted data by reflecting the result of convolution into some range. The regular Neural Networks adopt Sigmoid and tanh function as the activation function. Sigmoid function is given by Equation 3.1, which restrains the output value into the range of $[0, 1]$. Tanh function (also well known as hyperbolic function) is given by Equation 3.2, which limits the value to the range of $[-1, 1]$. As these two functions map large ranges of the input space to a small range, it is likely that a large change in the input produces a small change in the output and thus small gradient. The gradient vanishes as more activation layers stacked.

$$\sigma(x) = (1 + e^{-x})^{-1} \tag{3.1}$$

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.2}$$

CNN uses Rectified Linear Units (ReLU) to avoid gradient vanishing. The formula derivation of ReLU is given by Equation 3.3, where $\sigma(.)$ is sigmoid function. $\sum_{i=1}^{\infty} \sigma(x - i + 0.5)$ is referred as stepped sigmoid units and $\log(1+e^{x})$ is the softplus function. The softplus function can then be approximated by $\max(0, x)$, known as ReLU activation function. As

[2]Some literatures call use kernels to represent the filters.

shown in Figure 3.1, sigmoid and tanh functions squash the input space into a small range, while ReLU maps the convolution result to the range $[0, +\infty]$.

$$f(x) = \sum_{i=1}^{\infty} \sigma(x - i + 0.5) \approx \log(1 + e^x) \approx \max(0, x) \tag{3.3}$$

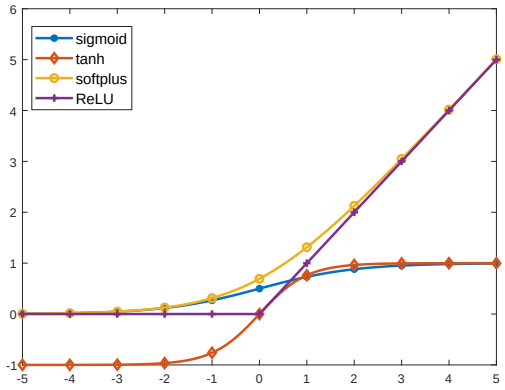


Figure 3.1: Activation functions comparison.

Pooling Layer

Pooling layer is inserted periodically in-between successive convolutional layers in a common CNN architecture. By representing multiple values in a block using a single value, pooling layer is also known as down-sampling or sub-sampling layer. The max-pooling uses the maximum value in the block to represent the block. The average-pooling calculates the average value of the block to represent the block. The use of pooling layer helps to reduce the number of parameters and control overfitting.

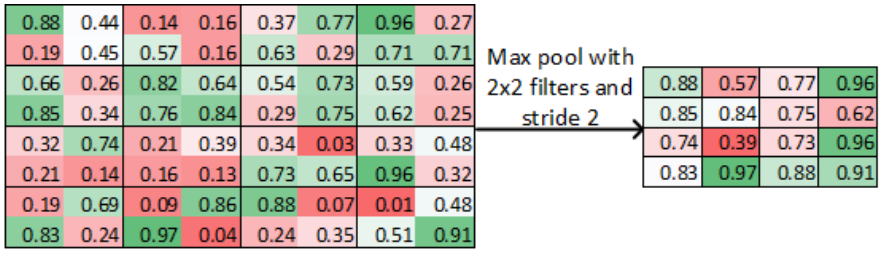


Figure 3.2: Example of a max pooling layer.

The pooling layer works by dividing the spatial input volume into several blocks with the same size. Hyperparameters of the pooling layer decide the size of the block. As shown in Figure 3.2 where the width, height, and striding size are all set as 2, the pooling result discards 3/4 of the input values. Generally, for an input volume of size $W \times H \times D_{in}$ and pooling filter of size F and stride S, the size of the output volume is given by $(W-F)/S+1$ and $(H - F)/S + 1$ with depth unchanged.

Fully Connected Layer

Neurons in fully connected layers are connected to all nodes in the input volume. Fully connected layers can be considered as a special convolutional layer with the filter and input volume of the same spatial size. Thus The spatial size of the output volume is then $(W - F + 2P)/S + 1 = (W - W + 0)/S + 1 = 1$. The output will be a vector of length K. The number of neurons, i.e., the value of K and the size of the input volume, controls the number of parameters in fully connected layers

3.1.2 AlexNet

AlexNet is the first commonly used CNN architecture and attracts close attention in research of CNN in computer vision. As shown in Figure 3.3 [73], the architecture of AlexNet contains eight layers, including five convolutional layers and three fully connected layers. Several sparkles are contributing to the success of AlexNet.

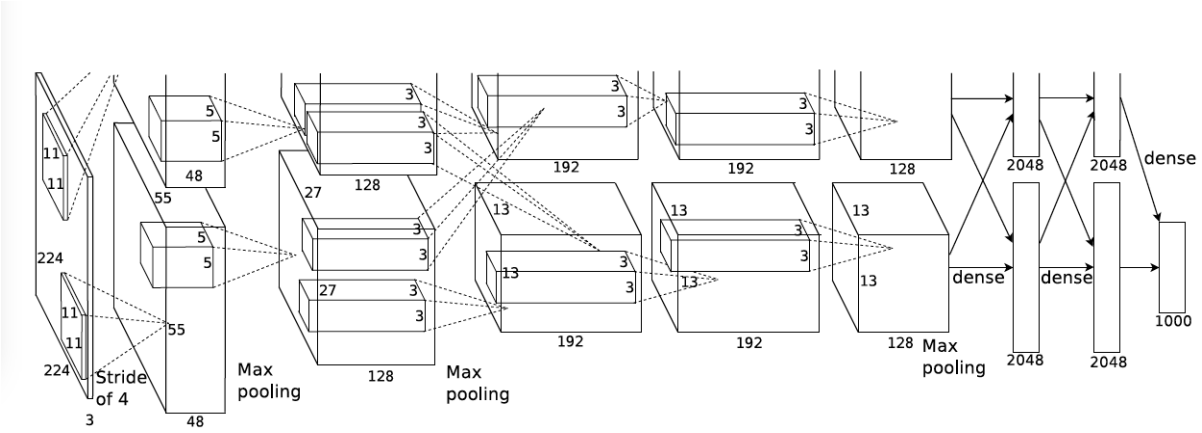


Figure 3.3: AlexNet model [73].

(1) ReLu activation function: AlexNet model the neuron's output using Rectified Linear Units (ReLU) due to its multiple advantages. Besides the benefit of avoiding gradient vanishing as mentioned in 3.1.1, the max function in ReLU introduces sparsity in the hidden units. Also, as it is frequent to compute the activation and its derivative in CNN, the piecewise linearity of ReLU makes it easy to implement the forwarding computation and partial derivative without division or exponential operating. The use of ReLU benefits from speeding up the forward and backward pass of AlexNet.

(2) Local Response Normalization: Local Response Normalization (LRN) is an extra layer following the convolutional and ReLU layer. It is also referred as *norm layer*. The response normalization $b^i_{x,y}$ is obtained by Equation 3.4, where $a^i_{x,y}$ is the output of

filter i at a particular position (x, y) after ReLU.

$$b^i_{x,y} = a^i_{x,y}/(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(K-1,i+n/2)} (a^j_{x,y})^2)^\beta \tag{3.4}$$

As a hyperparameter of the corresponding convolutional layer, K still denotes the number of filters. In Equation 3.4, the summation is performed over n filters applied to the same position along the depth of the input volume. Specifically, it sums the $n/2$ square of input along depth before j (the minimum value is 0) and $n/2$ square of input after j (the maximum value is $K-1$). The constants k, n, α and β are hyperparameters belong to LRN layer. AlexNet sets the hyperparameters as $k = 2, n = 5, \alpha = 10^4$, and $\beta = 0.75$[3]. Using response normalization, AlexNet achieves lower error rate than the version without normalization.

(3) Overlapping pooling: Different from the example in section 3.1.1 which takes a pooling filter of size 2×2 and stride 2, AlexNet configures the size of the filter larger than the stride. As a result, the neighborhoods summarized by adjacent pooling units will be overlapped. Overlapping pooling also helps to reduce the error rate.

(4) Data augmentation: AlexNet employs two types of data augmentation to enlarge the dataset to combat overfitting on image data. The first one is to generate image translations and horizontal reflections. The other way is to alter the intensities of the RGB channels in training images. Data augmentation is particularly helpful when the dataset is not large enough.

(5) Dropout: *Dropout* technique sets the output of each hidden neuron to zero with probability 0.5. The dropped neurons do not contribute to the forward pass and do not participate in backpropagation. The use of dropout forces the network to learn more robust features. Dropout is used in the first two fully-connected layers. With dropout, AlexNet reduces substantial overfitting.

In summary, the use of LRN and overlapping pooling contributes to reducing error rate; the techniques of data augmentation and dropout contribute to combat overfitting. The ReLU activation makes the training speed faster. ReLU has been widely and commonly used in CNN architectures. As the winner of ILSVRC 2012, AlexNet has been well pre-trained on the benchmark dataset of ImageNet which includes 1.2 million images with 1,000 object categories.

3.1.3 ZFNet

Zeiler and Fergus explored how to improve the performance of AlexNet and proposed ZFNet [126]. ZFNet reduces size of the filter in the first convolutional layer from 11×11 to 7×7 filters and changes the stride from 4 to 2.

[3] n denotes the size of the normalization neighborhood.

Table 3.1: Hyperparameters comparison between AlexNet and ZFNet

AlexNet	**ZFNet**
input image($\mathbf{227 \times 227 \times 3}$)	input image($\mathbf{224 \times 224 \times 3}$)
Conv1: 96, $\mathbf{11 \times 11}$, $\mathbf{stride = 4}$, $\mathbf{pad = 0}$ ReLu1 Norm1 Pool1: max, stride=2, $\mathbf{pad = 0}$	Conv1: 96, $\mathbf{7 \times 7}$, $\mathbf{stride = 2}$, $\mathbf{pad = 3}$ ReLu1 Norm1 Pool1:max, stride=2, $\mathbf{pad = 1}$
Conv2: 256, 5×5, $\mathbf{stride = 1}$, $pad = 2$ ReLu2 Norm2 Pool2: max, stride=2, $\mathbf{pad = 0}$	Conv2: 256, 5×5, $\mathbf{stride = 2}$, $pad = 2$ ReLu2 Norm2 Pool2:max, stride=2, $\mathbf{pad = 1}$
Conv3:384, 3×3, $stride = 1$, $pad = 1$ ReLu3	Conv3:384, 3×3, $stride = 1$, $pad = 1$ ReLu3
Conv4:384, 3×3, $stride = 1$, $pad = 1$ ReLu4	Conv4:384, 3×3, $stride = 1$, $pad = 1$ ReLu4
Conv5:384, 3×3, $stride = 1$, $pad = 1$ ReLu5 Pool5: max, stride=2, $\mathbf{pad = 0}$	Conv5:384, 3×3, $stride = 1$, $pad = 1$ ReLu5 Pool5: max, stride=2, $\mathbf{pad = 1}$
Fc6: 4096 ReLu6 Drop6 50%	Fc6: 4096 ReLu6 Drop6 50%
Fc7: 4096 ReLu7 Drop7: 50%	Fc7: 4096 ReLu7 Drop7: 50%
Fc8: 1000	Fc8: 1000

The hyperparameters of AlexNet and ZFNet are shown in Table 3.1. The size of input image for these two network are different. AlexNet receives $227 \times 227 \times 3$ image input while ZFNet takes $224 \times 224 \times 3$. ZFNet uses zero-padding in the first and second layer to keep the border information of the image. Furthermore, finer local features can be captured from the input volume with a smaller filter and stride. By tweaking several hyperparameters of AlexNet architecture, ZFNet reduces the top-5 error by 1.7%.

As stated in section 3.1.1, size of the processed volume is changing layer by layer. According to the hyperparameters in Table 3.1, we can evaluate the size of output volume for each layer [118]. The output size changing for AlexNet and ZFNet are shown in Figure 3.4 and Figure 3.5 respectively. As ReLU and norm layer do not change the size of input volume, we neglect the ReLU and norm layers in size track. AlexNet completes the computation by separating convolutional filters of each layer into 2 GPU groups. The output volume in ZFNet changes along convolutional and fully-connected layers with no grouping. It should be noted that the size of the final feature map before the last max pooling is about 17 ($227/13 \approx 17$) times smaller than the original image.

AlexNet and ZFNet are classical and important CNN architectures. VGG network and other extensions are based on AlexNet structure. As stacking more layers implies more

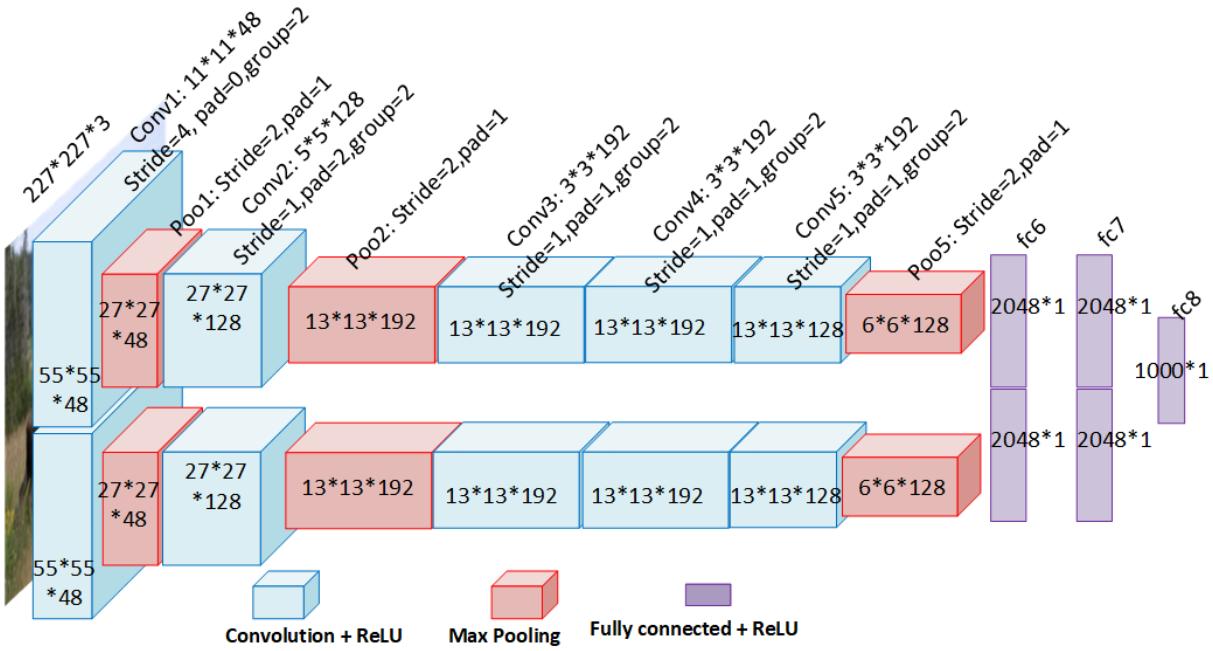


Figure 3.4: Output size changing among AlexNet layers.

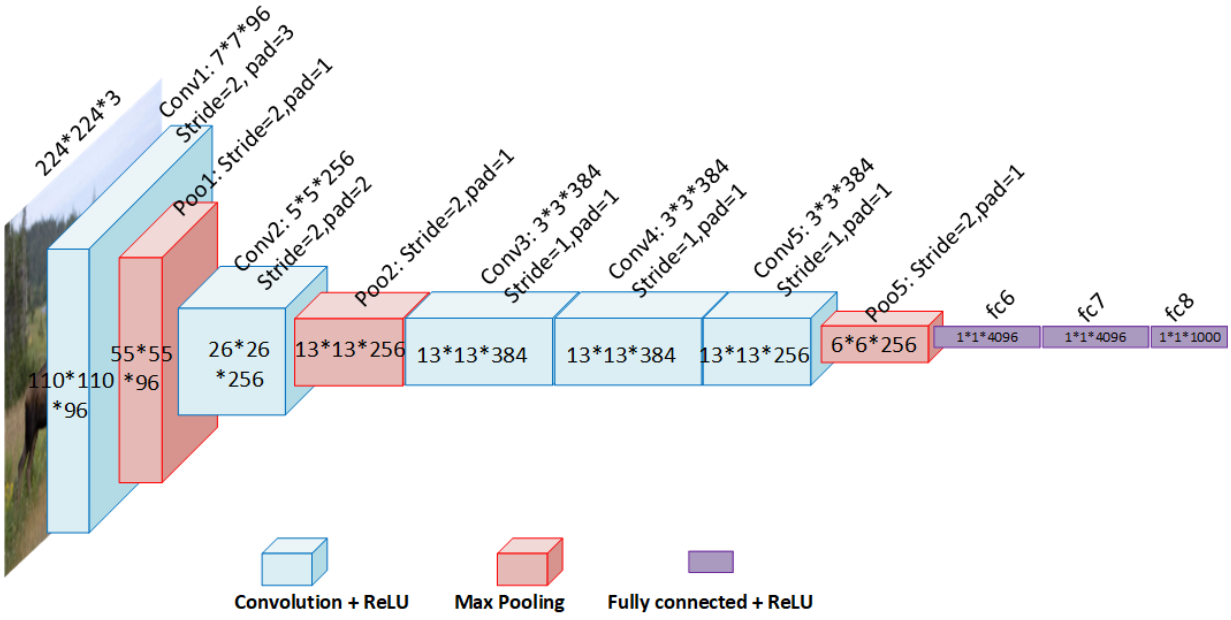


Figure 3.5: Output size changing among ZFNet layers.

GPU memory and GPU computation, one of the compromising methods for practical application is to use an architecture with fewer layers. This thesis employs ZFNet as the network for classification.

3.1.4 Feature Learned from CNN

The network model of CNN learns abstract features from image. The convolutional part of the network constructs a hierarchical representation of the input image. Deeper layers are able to extract higher-level features. To observe the features learned from the CNN network, we visualize the activation map resulted from the pre-trained AlexNet.

Figure 3.6: All 96 feature map of conv1.

As stated, the feature map resulted from the first convolutional+ReLU layer of AlexNet is of size $55 * 55 * 96$, i.e, a total of 96 spatial feature maps. All 96 feature maps from the first convolutional+ReLU layer are shown in arrays of Figure 3.6. Some of the maps still has bright intensity to show, while others are weak to recognize by human eyes. Different channels of feature map highlight distinguished characteristics of the moose, which can be explicitly told from Figure 3.7b and 3.7c.

(a) (b) (c)

Figure 3.7: Original image and feature maps of conv1. (a) Original image. (b) Feature map from channel 32. (c) Feature map from channel 85.

Discovering the feature map from deeper layer, we can observe more abstract features. As shown in Figure 3.8, the feature maps of the fifth convolutional+ReLU layer describe

complicated but special characteristics for the original image. As shown in Figure 3.8a, the contour of the moose is highlighted. In contrast, the body position of the moose is highlighted in Figure 3.8b. Through the visualization, it is convincing that abstract features can be extracted from CNN network.

(a) (b)

Figure 3.8: Feature maps from the fifth convolutional layer. (a) Feature map from channel 30. (b) Feature map from channel 68.

3.2 Object Detection with CNN

More than just recognition, object detection aims to localize each object in an appropriate position. The localization is represented by drawing a bounding box around the object. Therefore, the architecture of object detection is more complicated than that of image classification. In this section, we elaborate the frameworks of R-CNN family which combine region proposal with CNN.

3.2.1 R-CNN

The Region-based Convolutional Neural Networks (R-CNN) boost new progress in the research of object detection by adopting CNN as feature extractor. As shown in Figure 3.9, R-CNN consists of four main steps: region detection, features extraction, classification and bounding box regression [57].

1. **Region Proposal**: Candidate regions are generated through region proposal. The proposed regions are then cropped and warped for the following process.
2. **Feature Extraction**: Each proposed region is fed into a well-modeled CNN to extract CNN features.
3. **Classification**: The SVM classifier labels the CNN features.
4. **Regression**: The linear regressor refines the bounding boxes whose corresponding features are labeled as objects.

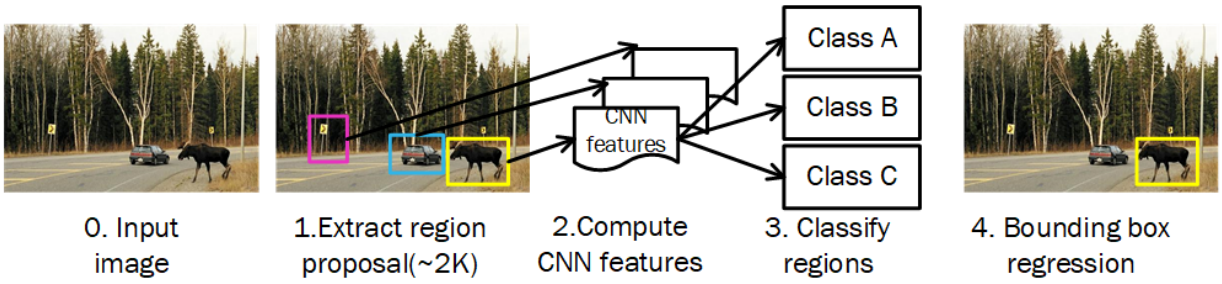


Figure 3.9: frameworks of R-CNN

Region Proposal

R-CNN adopts Selective Search to generate ROIs. With initial regions obtained through graph-based segmentation [52], Selective Search measures the similarity between regions using complementary measurements. Specifically, color histogram, texture histogram, the ranking of size and the fitting property of two regions are all considered. By merging two regions of highest similarity iteratively, selective search proposes a small set of data-driven, class-independent, high-quality locations. Appendix C can be referred to for the detailed algorithm.

In this thesis, we implement R-CNN and Fast R-CNN using Edge Box as the region proposal method. Edge Box method is open source for application and outperforms Selective Search [131]. Edge Box scores the candidate windows through the number of edges wholly enclosed in the region. Edge Box generates regions using efficient data structure and smart search strategies. The method searches for candidate regions using sliding window over position, scale, and ratio for further scoring. The scoring strategy is based on edge-response map by the Structured Edge detector [43, 45, 46]. *Edge groups* are computed from the edge-response map by gathering the edge points which are approximately on a line. The proposed regions are scored by accumulating the weights, which denotes whether an edge is wholly enclosed in the region. In practice, an integral image is used to speed up the computation. The detailed algorithm can be referred to Appendix B.

Feature Extraction

In R-CNN, the proposed regions need to be warped to a fixed size to fit the CNN input. As shown in 3.1, the fixed size of AlexNet and ZFNet are $227 \times 227 \times 3$ and $224 \times 224 \times 3$ respectively. Each proposed region should be resized to the fixed size accordingly. The warping step is followed by feature extraction using a CNN network similar to that in image classification. As in AlexNet and ZFNet, the final feature vectors from the first and second fully-connected layer are of size 1×4096. It's noteworthy that R-CNN needs to extract CNN features from each warped region and then classify them separately. For an image where 2,000 ROIs are generated, CNN should be performed 2,000 times and then output 2,000 CNN features. ,

Classification

The resulted CNN feature vectors are then fed into a linear classifier for further classification. R-CNN uses SVM classifier to map the features to category scores. SVM uses a soft margin to separate many but not all data points. SVM classifies data by finding the optimal hyperplane that separates data of one category from the other. In case that the data points do not allow for a separating hyperplane, SVM uses a soft margin to separate many but not all data points. A hinge loss function is used to make a soft margin in SVM.

The classifier is trained through back propagation method using training data. In the training step of the R-CNN detection, CNN feature vectors extracted from the ground-truth bounding box are used as positive training data, while feature vectors from regions whose Intersection over Union (IoU) with the ground-truth ones are less than 0.3 as negative training data [58].

Bounding Box Regression

There is an extra Non-Maximum Suppression (NMS) step at test time. After the extracted feature vector from each region has been scored using the pre-trained classifier, the NMS reject lower-scored regions greedily. Generally, among overlapping regions which overlap with each other at a higher ratio than a user-defined threshold, only the region with the highest score is reserved.

The step of bounding box regression fine tunes the bounding boxes for the qualified region by well-trained parameters to better localize the object. The regression is implemented through a linear ridge regressor, which is trained by a set of pairs $(P^i, G^i)_{i=1,\ldots,N}$ [51]. Specifically, $P^i = (P^i_x, P^i_y, P^i_w, P^i_h)$ denotes the raw bounding box of the proposal P^i by the pixel coordinates of the center (P^i_x, P^i_y) and the width P^i_w and height P^i_h. G^i denotes the ground-truth bounding box in the same way.

In order to map a proposed box P to a ground-truth bounding box G, the regressor is trained to optimize the transformation in four functions $d_x(P)$, $d_y(P)$, $d_w(P)$ and $d_h(P)$, specifying translation of the center coordinates as well as the log-space translation of the width and height. The predicted bounding box $\hat{G}$ can be obtained by applying the transformation in Equation 3.5.

$$\begin{aligned} \hat{G}_x &= P_w d_x(P) + P_x \\ \hat{G}_y &= P_h d_y(P) + P_x \\ \hat{G}_w &= P_w \exp d_w(P) \\ \hat{G}_h &= P_h \exp d_h(P) \end{aligned} \tag{3.5}$$

The four functions are modeled by feature maps of the proposed box P from the last convolutional layer of the CNN network. For AlexNet, the feature map is the output of fifth convolutional layer after ReLU and max pooling, denoted by $\phi_5(P)$. Generally, $d_*(P)$ where $*$ is one of x, y, w, h is a linear function of $\phi_5(P)$. Denoting the learnable

parameter vector using w_*, the linearity is given by Equation 3.6. w_* can be learned by the optimization solution of ridge regression.

$$d_*(P) = w_*^T \phi_5(P) \tag{3.6}$$

Combining the bottom-up region proposals and convolutional networks, R-CNN improves the mean Average Precision (mAP) over the dataset of Visual Object Classes (VOC) 2007 to 58%. However, R-CNN resizes the proposed regions to a fixed size and brings about undesirable deformation. Moreover, the vast number of calling on CNN slows down the detection.

3.2.2 Fast R-CNN

Spatial Pyramid Pooling (SPP) is introduced to avoid region warping [63]. As shown in 3.10, an SPP layer is inserted between convolutional layers and fully connected layers. Instead of cropping and warping the proposed regions for numerous feature maps, SPP-Net obtains a single feature map and utilizes SPP layer to extract a fixed-size feature representation for each region.

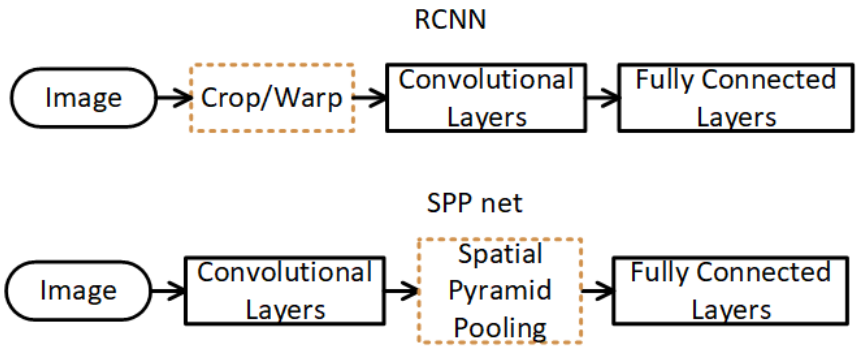


Figure 3.10: Framework of R-CNN (top) and SPPNet(bottom).

As shown in Figure 3.11, the CNN network obtains feature maps of different spatial size with an input image of arbitrary size. The SPP layer divides each feature map into 1×1, 2×2 and 4×4 blocks and pools out the maximum value of each block. The output of the SPP layer is a fixed-length vector which concatenates the maximum values in order. The final length is $(16 + 4 + 1)K = 21K$, where K denotes the depth of feature maps. Using different block size may alter the length of the feature vector.

Fast R-CNN adopts ROI pooling layer [56], as shown in Figure 3.12. The convolutional layers are performed once to calculate the feature map. We reflect the proposed regions from original image to the bounding box on the feature map. The ROI pooling layer then extracts fixed-size representation from each corresponding bounding box. Another improvement in fast R-CNN is replacing SVM classification with the Softmax layer which extends the structure of neural network for better predictions.

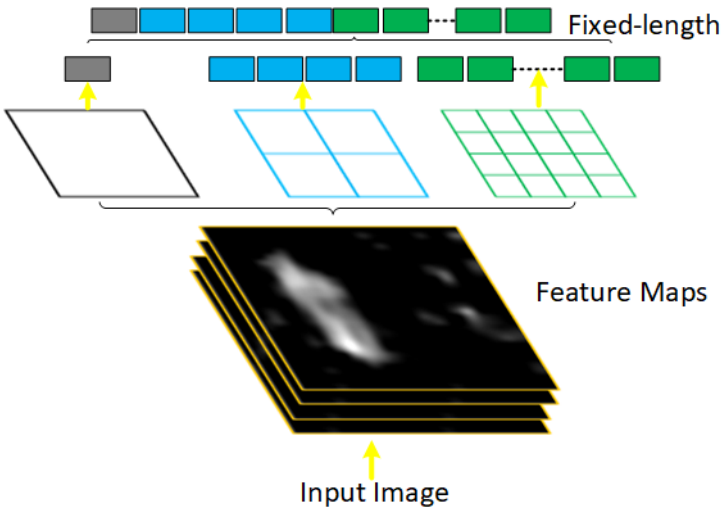


Figure 3.11: Spacial Pyramid Pooling layer

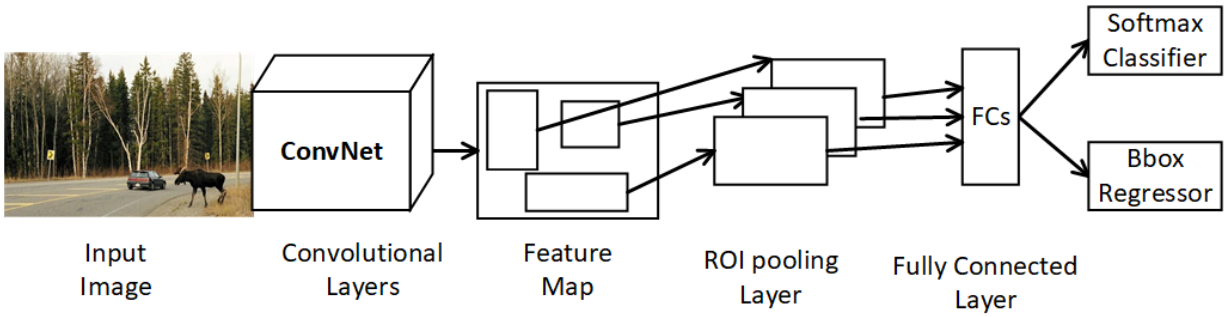


Figure 3.12: Framework of fast R-CNN.

The ROI pooling layer

The ROI pooling layer converts the feature map into a number of feature vectors of fixed size $H \times W$. The size arguments H and W of the output vector are predefined layer hyperparameters. We denote the proposed ROI by a tuple (x_0, y_0, h, w), where (x_0, y_0) denotes the left-head corner of the bounding box, h and w defines the height and width. The ROI pooling layer divides the feature map of size $h \times w$ into blocks of size $H \times W$ and pools the maximum value or average value out to represent each block. The amount of blocks a round value of $h/H \times w/W$.

The training and testing procedures of fast R-CNN achieve faster speed due to the shared computation of feature map. Besides, by replacing the SVM classifier in R-CNN with a single Softmax layer, single-stage training of Fast R-CNN can be implemented using a multi-task loss. As a result, Fast R-CNN gains better accuracy as the mAP on the dataset of VOC 2007 is raised to 68%.

3.2.3 Faster R-CNN

Fast R-CNN continues to use Selective Search as region proposal method. Faster R-CNN breaks the bottleneck in the process of region proposal by generating candidate regions using CNN—the Region Proposal Network (RPN) [101]. As shown in Figure 3.13, feature maps are obtained from a series of convolutional and ReLU layers. RPN works on the feature map through two sibling layers, a Softmax classifier and a bounding box regressor. The output of RPN is then fed into the following CNN structure for finer classification and bounding box regression.

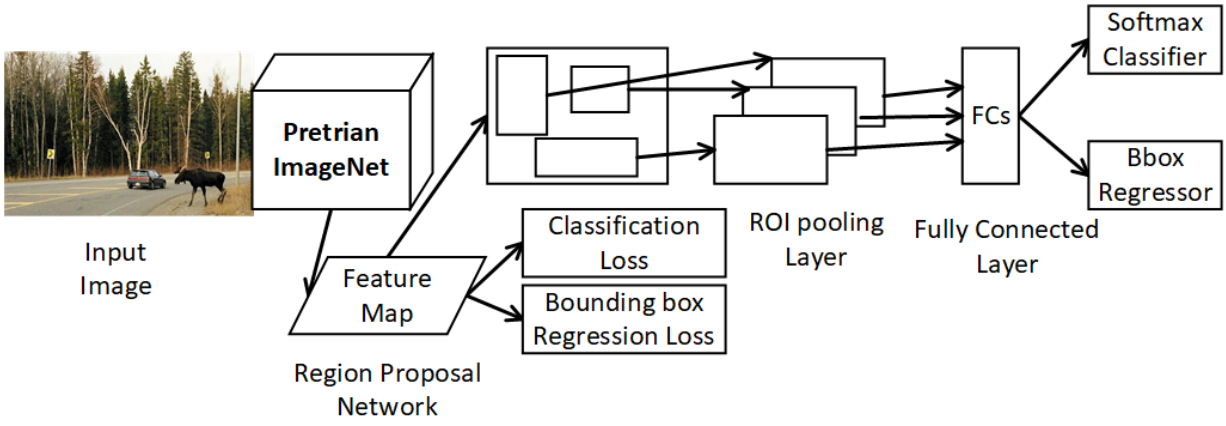


Figure 3.13: Framework of faster R-CNN

To be specific, RPN segments the image through window-sliding on the feature map. As shown in Figure 3.14, the 3×3 window is slided across the feature map just like the convolution operation by a filter of the same size. For each 3×3 sliding-window location, multiple reference boxes with different scale and ratio are proposed. Each reference box is called an *anchor*. Nine different anchors are generated by three scales (1/2, 1, 2 times original region's area) and three aspect ratios ($1:1,\ 1:2,\ 2:1$). Thus $k = 9$ in Figure 3.14.

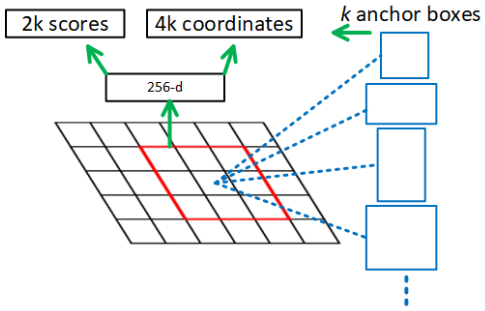


Figure 3.14: region proposal network

In the example of ZFNet, the resulted 256 feature maps are of the size 13×13 and

there are totally $13 \times 13 \times 9$ anchors. A feature vector of 256-dimension is obtained for each 3×3 window in ZFNet by convolving with filter of depth 256. After the intermediate layer, a Softmax classification layer and a bounding box regression layer for k anchors extend the network. The classification layer outputs $2k$ scores representing the probability of being a target or background for all anchors. The regression layer outputs $4k$ parameters representing the transformation on coordinates of left-top corner, height and width of the raw bounding box. These two sibling layers are constructed by fully connection with the former 256d feature vector. In terms of convolution, these two layers could be considered as a special convolution with kernel of size $1 \times 1 \times 18$ and $1 \times 1 \times 36$ respectively.

RPN propose regions in the manner of sliding window so that the regions can cover all positions on the image. Additionally, 9 variant anchor boxes are generated based on one box, the proposed regions are invariant to translation, ratio and scale. RPN and the following fast R-CNN share convolutional features so that the region proposal step is nearly cost-free. Using the structure of VGGNet as feature extractor, faster R-CNN gains higher mAP up to 73% on VOC 2007.

3.2.4 Summary

Based on the CNN architecture, we introduce the structure of the R-CNN family for object detection. Using region proposal method to generate ROIs, R-CNN employs CNN to extract feature and uses SVM to classify the regions. By further post-processing the resulted bounding boxes, R-CNN improves the accuracy of object detection. Fast R-CNN gains speed-up by inserting ROI pooling layer. Faster R-CNN implements region proposal using RPN and achieves faster speed. Based on these techniques, we propose to apply a novel region proposal method in the framework of Fast R-CNN.

3.3 Maximally Stable Extremal Regions

In R-CNN and Fast R-CNN, the approach of region proposal plays an important role. The existing methods, such as Selective Search and Edge Box, generate a large number of candidate regions. The huge amount of proposed ROIs affects the processing efficiency of the entire detection system. In this section, we introduce a novel method of region proposal using Maximally Stable Extremal Regions (MSER) to reduce the number of proposals.

3.3.1 MSER Algorithm

For a region, we denote the arbitrary pixel inside as p and corresponding intensity $I(p)$. We then denote the arbitrary pixel on the boundary of this region as q and intensity $I(q)$. If the region and its boundary hold $I(p) > I(q)$, this region is called a maximum intensity region or maximal region. If $I(p) < I(q)$ hold, it is a minimum intensity region or minimal region [84].

The computation of extremal regions can be considered as a process of gray image binarization with changing threshold. For a gray image I, the binarized image is all white when the threshold t is 0. As we increase the value of t, the black spots in the binarized result appear and grow, as shown in Figure 3.15. The growing black spots correspond to maximal regions. As the spot dilates, regions corresponding to two maximal local spots will merge at some point. In other words, multiple components can be connected at some threshold level. Finally, a black image is obtained with all regions merging into one or all components connected into one component. The set of all connected components of the image is the set of all maximal regions. Similarly, minimal regions can be obtained by inverting the intensity of I and running the same process. In a word, every extremal region is a connected component of a thresholded image.

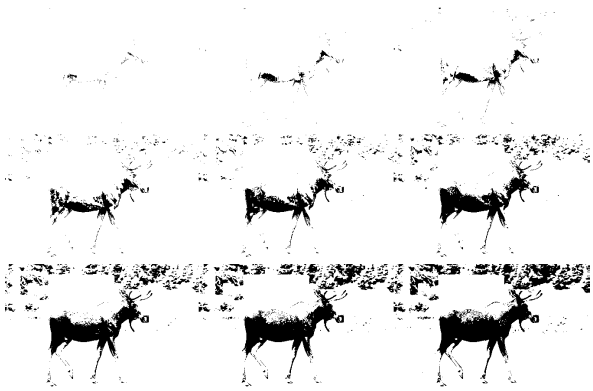


Figure 3.15: The process of image binarization when changing threshold.

For an image, a sequence of nested extremal regions $\mathcal{Q}_1, ..., \mathcal{Q}_{i-1}, \mathcal{Q}_i, ...$ are obtained with $\mathcal{Q}_{i-1} \subset \mathcal{Q}_i$. Extremal region $\mathcal{Q}_i$ is Maximally Stable Extremal Regions (MSER) if and only if $|\mathcal{Q}_{i+\Delta} - \mathcal{Q}_{i-\Delta}|/|\mathcal{Q}_i|$ has a local minimum at i, where Δ is the alternating value of the threshold.

Matas kept tracks of the extremal regions using union-find data structure with path-compression [84]. The computational complexity was $\mathcal{O}(\log \log n)$, i.e. almost linear, where n is the number of pixels. D. Nister *et al.* proposed new method to implement MSER in true linear time $\mathcal{O}(n)$ [91].

Regarding the image intensity profile as a territory with different landscapes as shown in Figure 3.16, we seek for the extremal regions by filling water to the whole territory. One area filled with water denotes an extremal region. In standard MSER computation, the filling happens like it is raining evenly all over the territory. The water level is rising to all areas at the same time. In linear MSER implementation, the filling occurs as if it is showering locally in one initial place. As the local basin area is flooded, water overflows to the neighboring basin, and then the next area starts filling up. Smaller regions of water join and become bigger bodies of water, and the whole area gets filled finally. As water

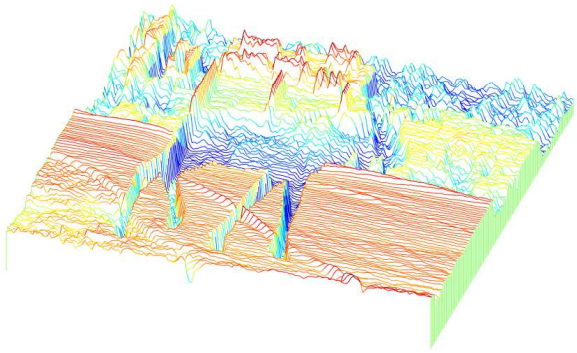


Figure 3.16: MSER watershed

is filling up into a local area, the extremal region is checked against the MSER stability criterion.

The linear MSER algorithm makes use of stack and queue to store connected component and boundary pixels for greedy searching. The detailed algorithm can be found in Appendix A.

MSERs are considered as feature regions since MSER regions are invariant to affine transformation of image intensities and covariant to adjacency preservation. Besides, MSER achieves a high degree of invariance to illumination and nearby cluttering. Additionally, maximally stable extremal regions remain unchanged over a wide range of selected thresholds. The scale invariance is easily improved by computing MSER over a scale pyramid and removing duplicate detections [54]. Due to these advantages, MSER becomes commonly used in image retrieval, recognition and tracking tasks.

3.3.2 MSER Regions

As the component evolves in the component tree[4], the component information can be tracked. The information could be in the form of a linked list including pixels of the component. Denoting the pixel of the component by its coordinate (x, y), the linked list is a connection of these coordinates. The component information could also consist of the first and second order moments of the regions.

With these moment values, we can turn the MSER into elliptical regions. As shown in Figure 3.17a, the pixels of an extracted moose region are highlighted. Figure 3.17b fits the highlighted component to an ellipse. Figure 3.17c displays the ellipse only. The ellipses which have the same second moments with the MSER regions can properly fit the majority of the region. The purpose of fitting the regions to geometry ellipses is to approximate

[4]Data structure tree is used to store the component in MSER algorithm

(a) (b) (c)

Figure 3.17: MSER region and its fitted ellipse. (a) An MSER region. (b) The MSER in (a) is fitted to an ellipse. (c) The fitted ellipse only.

the shape of the complexed region with a simple shape. In chapter 4, we discuss how to propose candidate regions from both pixel list and ellipses.

3.3.3 Summary

In this section, we introduce MSER region detection. Using watershed algorithm, MSER can be implemented in linear time. The output of MSER has two forms. One form is a linked list of pixels in the region. The other form is a collection of structure representing the corresponding ellipses. In this thesis, we employ MSER as a novel region proposal method and apply it in R-CNN and Fast R-CNN detection architecture. To generate bounding boxes of animals, both forms of MSER output can be used. The detailed application of both MSER output forms are explained in Chapter 4.

3.4 Measurement

There are measurements for both image classification and object localization. As for image classification, we measure the accuracy of the classifier to evaluate the ability to distinguish target image from non-target. As for object localization, we assess the detector by the ability to position the correct target.

3.4.1 Overlap Ratio

The overlapping ratio is calculated to measure the localization accuracy of the detected bounding box. The most commonly used ratio is Intersection over Union (IoU). As shown in 3.18, two bounding boxes are intersected, where one represents the ground-truth bounding box and the other denotes the detected bounding box. IoU is defined in Equation 3.7 by

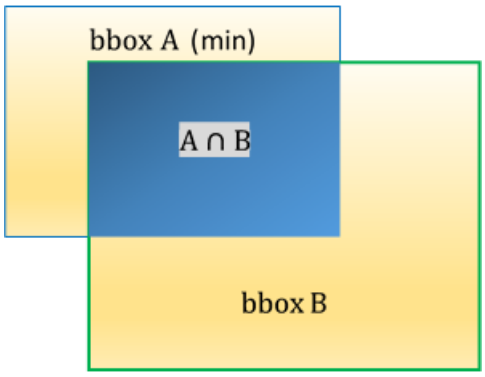


Figure 3.18: Overlap ratio of bounding box A and bounding box B

the overlapping area of A and B to their union area [57].

$$\begin{aligned} IoU &= \frac{area(A \cap B)}{area(A \cup B)} \\ &= \frac{area(A \cap B)}{area(A) + area(B) - area(A \cap B)} \end{aligned} \tag{3.7}$$

Another measurement Intersection over Minimum (IoM) is defined in Equation 3.8 as the rate of intersection area between A and B to the minimum area of the two bounding boxes. As shown in Figure 3.18, bounding box A takes a smaller area, the IoM is then $area(A \cap B)/area(A)$.

$$IoM = \frac{area(A \cap B)}{\min(area(A), area(B))} \tag{3.8}$$

IoU is used when evaluating the performance of localization. A higher IoU denotes that the detected bounding box is closer to the expected one. As shown in Figure 3.19, the two green rectangles are the expected positions of the target while the yellow rectangle is the detection result. The yellow box has an IoU of 0.8263 with the left green bounding box while 0.3034 with the green bounding box to the right. Namely, the detection to the large moose is better than that to the small moose.

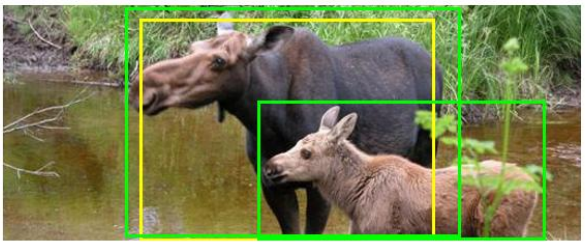


Figure 3.19: How good is the detection

When evaluating the performance of a detection system based on recognition, we match the detected bounding box to a ground truth one if they overlap sufficiently. Specifically, we take the PASCAL measure, which states that the overlap ratio must exceed 50% [66]. Thus, the threshold of IoU is set as 50% as an acceptable detection in this paper. Based on this threshold, the detection results in Figure 3.19 detect the large moose successfully while miss the detection of the small moose.

3.4.2 Confusion Matrix

We take an example of image recognition as an entry point. There exists an image dataset of size 200 containing 100 animals and 100 non-animals. A particular classifier might classify 90 of the 100 positive images as animal and classify 50 of the negative images as non-animal. Then the *true positive* (T_P) is $90/100 = 0.9$ representing the ability of correctly labeling positive candidates. The *false positive* (F_P) is $(100 - 90)/100 = 0.1$ which tells the rate of wrongly labeled positive candidates. Similarly, the *true negative* (T_P) is $50/100 = 0.5$ and the *false negative* (F_P) is $(100 - 50)/100 = 0.5$ denoting the rate of classifying negative images correctly and wrong respectively. The four rates are listed in the form of a table shown in Table 3.2.

Table 3.2: Confusion Matrix

	Animal	Non-animal
Animal	90 True Positives	10 False Negatives
Non-animal	50 False Positives	50 True Negatives

For this example, the detection accuracy $(90+50)/200 = 70\%$ denotes that 70% of both positive and negative images are correctly labeled. But in a more detailed perspective, the accuracy yields misleading results as the recognition rates vary greatly in terms of positive images, which is 90%, or negative image, which is only 50%. Therefore, the multiplication of the diagonal percentages in Table 3.2, $90\% \times 50\% = 45\%$ denotes a more accurate and considerate measurement.

Table 3.2 is called *confusion matrix*, which is a specific table or matrix with n rows and n columns that reports the number of true positives, false positives, false negatives and true negatives, where n represents the number of categories. And the multiplication of the diagonal values provides a comprehensive measurement for the classification. In this thesis, $n = 2$ as we only separate animals from background regardless of other objects. Rows of the confusion matrix stand for the actual class and columns stand for the predicted class. The confusion matrix is commonly used to evaluate the performance of a classifier.

3.4.3 Precision Recall Curve

Based on the four elements in Table 3.2, we can define more measurements. Precision is used to measure the result relevancy, while recall is to measure how many truly relevant

results are returned. Precision(P) is defined as the number of true positives (T_p) over the number of true positives plus the number of false positives (F_p). Recall is defined as the number of true positives (T_p) over the number of true positives plus the number of false negatives (F_n).

$$P = \frac{T_p}{T_p + F_p} \tag{3.9}$$

$$R = \frac{T_p}{T_p + F_n} \tag{3.10}$$

The precision-recall (PR) curve shows the relationship between precision and recalls for a different threshold. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall). In the calculation process, a detection with IoU higher than a predefined threshold is treated as a T_p. The PR statistics is calculated following the standard of PASCAL VOC 2011 [65].

A system with high recall and low precision returns many results, but most of its predicted labels are incorrect when compared to the ground truth labels. In contrast, a system with high precision returns very few results, but most of its predicted labels are correct when compared to the ground truth labels. An ideal system with both high precision and recall will return a large enough number of results labeled correctly.

Average Precision (AP) is the standard PASCAL VOC measure for object detection. The calculation of AP is to accumulate the area of PR curve. It can be obtained every time a new positive sample is recalled. Following the standard of PASCAL VOC 2011, AP is defined as the weighted mean of precision achieved at each recall value. The weight is computed as the increase in recall from the previous. The calculation of AP is given by Equation 3.11, where P_i and R_i are the sorted precision and recall at the ith index.

$$AP = \sum_i (R_i - R_{i-1}) P_i \tag{3.11}$$

3.4.4 Log Miss Rate Curve

Another measurement of detection used in this thesis is Miss Rate (MR) with respect to False Positive Per Image (FPPI) [44]. We plot the curve in log domain to show the miss rate in a larger range. We obtain the miss rate by subtracting truth positive from 1. A lower curve denotes a higher truth positive and smaller miss rate. In the calculation process of miss rate, the FPPI is accumulated when a detection IoU is less than the predefined threshold.

3.4.5 Recall/IoU-Threshold curve

A functional region proposal should cover the regions of interest in the test image [66]. The curve of *Recall/IoU-Threshold* is used to evaluate the quality of extracted regions. It describes the trend of recall when the IoU threshold is raised from 50% to 90% and higher. The curve from a better region proposal method should cover a larger area and the decreasing rate of recall should be smaller when the IoU threshold is increased. To be noted, Recall/IoU-Threshold curve evaluates the performance of region proposals and does not stand for the performance of object detection.

Chapter 4

Proposed Architecture

Region proposal is an important step of R-CNN [57] and fast R-CNN [56]. Other than the approach of selective search or edge boxes used in these two object detection methods, we propose to extract Regions of Interest (ROI) through MSER. To evaluate the performance of region extraction method, we propose new architectures based on R-CNN and fast R-CNN frameworks by applying MSER to generate ROI.

4.1 Preprocessing of MSER regions

As stated in former sections, a number of stable connected components can be detected through MSER algorithm from the input image. Before performing the MSER algorithm to the original image, we need to set up some parameters which will affect the number of regions extracted from the input image.

(1) The threshold delta Δ, or step size represents the step of intensity threshold levels during the process of MSER computation. The value of Δ indicates through how many different gray levels does a region need to be stable to be considered maximally stable. The variation of extremal region is given by Equation 4.1, where R denotes area of the previously detected region and $R + \Delta$ is area of the region when intensity level is raised by Δ. A "maximally stable" region has a local minimum variation, thus higher stability than the regions from previous and former thresholding levels.

$$V = \frac{area(R + \Delta) - area(R)}{area(R)} \tag{4.1}$$

As shown in Figure 4.1, different threshold values are used for MSER detection. A small value of Δ results to more detailed regions, while a higher Δ contributes to obtain less regions with higher stability. In this thesis, Δ is set to 1.6 based on calibrating experience.

(a) (b)

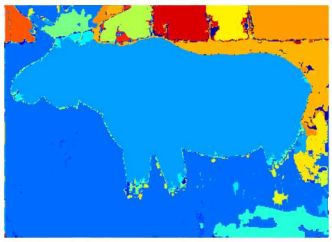
Figure 4.1: MSER threshold delta. (a) Resulted MSER regions with $\Delta = 1$. (b) Resulted MSER regions with $\Delta = 9$

(a) (b)

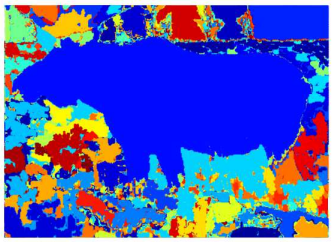
Figure 4.2: MSER area range delta. (a) Resulted MSER regions with area range [30 $0.15S$], where S is the area of the whole image. (b) Resulted MSER regions with area range [$0.15S$ $0.8S$]

(2) Area range, i.e. the size of the region in pixels, excludes the region whose area is beyond this range. As shown in Figure 4.2, changing the range of area leads to different results. In this thesis, the range is set according to the size of the image. For example, an object whose area size is beyond 90% of the whole tested image, or less than 5% would be rejected as it does not make sense to the actual scenario. Denoting the area of image with S, the actual range is set to $[40, 0.8S]$.

(3) Maximum area variation between extremal regions at varying intensity thresholds offers the restrict to stability of the results. That is to say, regions which are unstable would be rejected. Stable regions are very similar in size over varying intensity thresholds. Decreasing the value of this parameter helps to produce less but stable regions. As shown in Figure 4.3, a larger value of maximum variation provides more regions with various changing. Based on calibrating experience, the maximum area variation between extremal regions is set to 0.35.



(a)



(b)

Figure 4.3: MSER maximum area variation. (a) Resulted MSER regions with maximum variation 0.1. (b) Resulted MSER regions with maximum variation 0.8.

4.1.1 MSER regions

The goal of region proposal is to find coordinates of the ROIs that possibly contain animals from the input image. The output of MSER algorithm is a set of regions represented by a list of pixels. Optionally, ellipses can be calculated to represent the MSER regions. For the representation of pixel list, it is intuitive to achieve a minimum enclosing rectangle for all coordinates of the pixel list directly. For the representation of ellipses, we then need some calculations to obtain the bounding box of each ellipse.

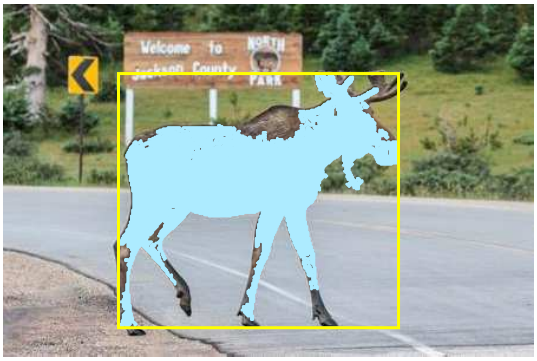
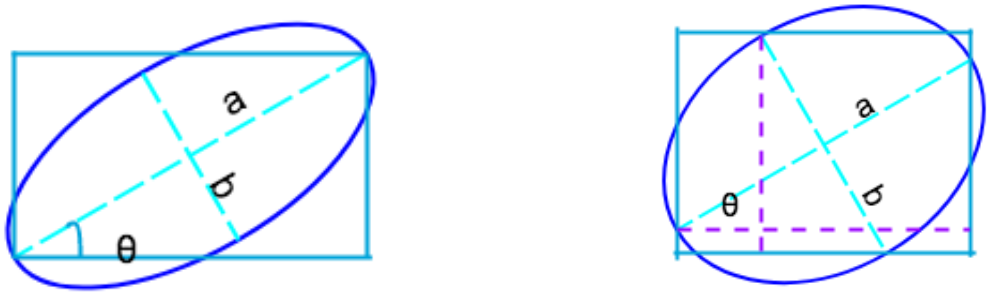


Figure 4.4: Bounding box for the pixel list of one MSER region.

Starting from different representation of the MSER region, we have two methods to find the coordinates of ROIs. For the first method, we seek for a minimum enclosing rectangle for pixels in the region. The pair of (x_{min}, y_{min}) and (x_{max}, y_{max}) can be easily obtained for the bounding box of the whole pixel list. (x_{min}, y_{min}) and (x_{max}, y_{max}) represent the minimum and maximum coordinate value in both horizontal and vertical axis. The resulted

bounding box is shown in Figure 4.4 where the yellow rectangle bounds the animal region.



(a) (b)

Figure 4.5: Fit an ellipse to its bounding box. (a) A slim ellipse and its bounding box. (b) A fat ellipse and its bounding box

For the second method, to obtain the ROI in form of bounding box, we need to find a bounding box for each MSER ellipse by geometrical transformation. As shown in Figure 4.5, we find a bounding rectangle which is orthogonal to the reference axes. The computation can be summarized by Equation4.2. The computed bounding box for the example region is shown in Figure 4.6a. Figure 4.6b displays the bounding box obtained from both methods.

$$width = \max(2a\cos\theta, 2b\sin\theta) \tag{4.2}$$

$$height = \max(2a\sin\theta, 2b\cos\theta) \tag{4.3}$$

The reason to use MSER ellipses to generate bounding box is that the extracted MSER regions often include extra pixels near the animal, especially when animals are blending with their surroundings. As shown in Figure 4.7, the blue bounding box is the enclosing rectangle for the resulted pixel list of the corresponding deer region. As the intensity of grass under the deer's feet is similar to that of the deer, extracted pixel list corresponding to the deer also contains some pixels of the grass. Thus, the bounding box resulted from the first method is larger than the deer as it encloses the strip of grasses on the ground inside. In contrast, the method of fitting the elliptical regions encloses the animal by centroids. As shown in Figure 4.7, the red bounding box represents the region of deer better by fitting to the approximate ellipse.

However, the rectangle fitting the ellipse is usually smaller than the MSER region. The reason is that when fitting the region to an ellipse, the ellipse with the same second moments just approximates the irregular area to its best but cannot describe the area

(a) (b)

Figure 4.6: MSER region and its fitting bbox by ellipse. (a) Bounding the ellipse with a rectangle. (b) Comparing two bounding boxes through pixel list and ellipse.

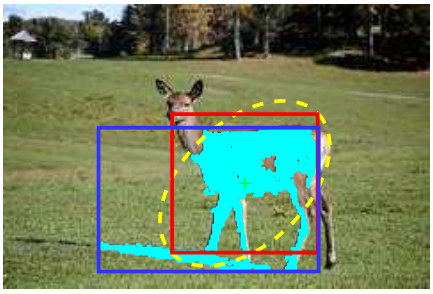


Figure 4.7: The fitted bounding box of the ellipse better encloses the deer.

perfectly. Based on this result, we need further steps to enlarge the bounding boxes. We use EL MSER to denote the method of acquiring bounding boxes from ellipses and PL MSER to denote the method of computing bounding boxes from pixel list.

4.1.2 Filtering out Undesirable Regions

In the actual implementation, the bounding boxes achieved from either method are not used as ROIs directly due to several causes. The first reason is that many ROIs are generated, where some ROIs are less likely to contain an animal. The amount of output regions has been reduced by the parameters of minimum and maximum areas, as well as the maximum variation, so that we reject ROI whose area is not within the defined range of area and is not stable enough. In this section, we are introducing another filter to further reduce the number of ROIs. Other reasons and dealing methods are discussed in later section.

The object ratio is often employed as an important feature in object detection. In the feature description of Deformable Part Model (DPM) [51], ratio is used to separate different components to detect objects in various perspectives. According to the observation to the appearance of 3,000 animals in the dataset, the aspect ratios of the corresponding bounding boxes are within a certain range. As shown in Figure 4.8, the width-height ratio is below 6 and most of the values fall in the range of $[1/3, 4]$. The height-width ratio[1] fall in a narrower range $[1/4, 3]$.

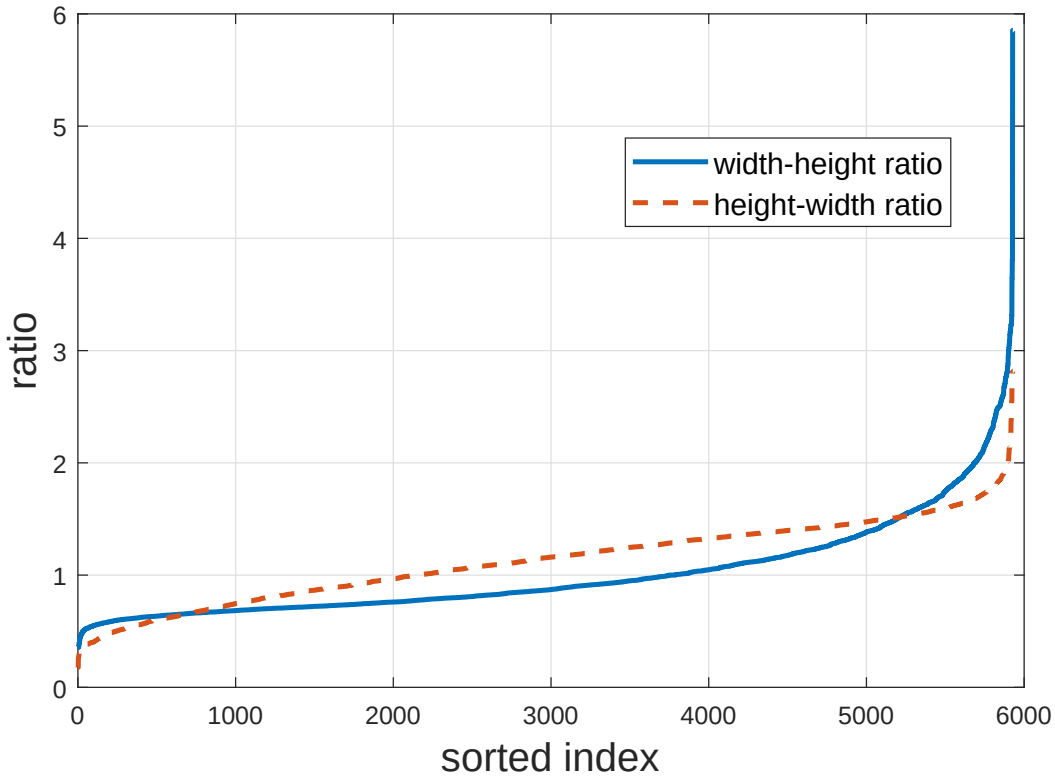


Figure 4.8: Sorted aspect ratios of animals' bounding box.

As an initial filtering step, this simple range can be used as a filter to keep regions whose aspect ratios fall between. Similarly, the ROIs whose aspect ratios fall outside are rejected by the simple filter. As shown in Figure 4.9, highlighted regions of the branch are rejected by the ratio filter.

Except for the simple range, we employ a probability distribution function to measure the ratio of extracted regions. To model the ratios, we take height-width ratio only as a random value and fit the observed ratio from the dataset to a mixed Gaussian model. Both width-height and height-width ratios can be reflected in the mixed Gaussian models.

Assuming that the ratios are drawn from the distribution of Equation 4.4, where k denotes the number of components, α_i is the proportion for each component of the mixed

[1]Height-width ratio is the reverse of width-height ratio

Figure 4.9: The highlighted regions are rejected by ratio filter

Gaussian distribution, μ_i and σ_i are mean and variance for the distribution respectively. Parameters of the Gaussian mixture model can be optimized to fit the actual distribution using the iterative Expectation-Maximization (EM) algorithm [89].

$$P(r) = \sum_{i=1}^{k} \alpha_i \mathcal{N}(\mu_i, \sigma_i) \tag{4.4}$$

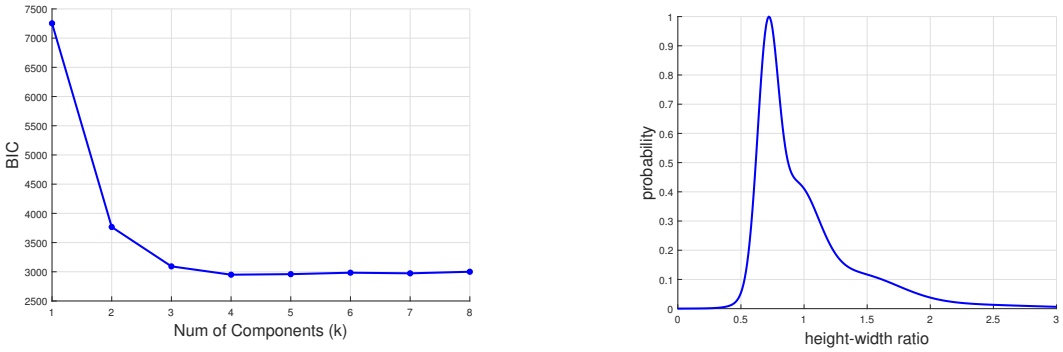
By changing the number of component k, we measure the fitted probability distribution by Bayesian Information Criterion (BIC) to find the best Gaussian mixture model [109]. BIC is given by Equation 4.5, where $NlogL$ is the negative log-likelihood, n is the number of observations, and m is the number of estimated parameters.

$$BIC = 2 \times NlogL + m \times log(n) \tag{4.5}$$

Model with the lowest BIC is preferred as a best model. As shown in Figure 4.10a, BIC changes with respect to the number of component k. The value of BIC falls to minimal when $k = 4$. Therefore, we use 4 components in the mixture Gaussian model. For each component, the corresponding parameters are shown in Table 4.1. With such parameters, the mixture Gaussian model is distributed as shown in Figure 4.10b. The most two powerful components in the mixture are when $i = 1$ and $i = 4$.

Table 4.1: Mixture Gaussian model parameters

parameters	α_i	μ_i	σ_i
i=1	0.379391	0.9286	0.0332
i=2	0.058778	2.0675	0.4909
i=3	0.208238	1.3894	0.1195
i=4	0.353594	0.7111	0.0065



(a) (b)

Figure 4.10: Mixture Gaussian models. (a) BIC trend over number of components. (b) The mixed Gaussian distribution with 4 components of height-width ratio.

The probability model is also used to score the proposed regions in later steps. Additionally, the ratio corresponding to the maximum probability value can be employed as a reference ratio for modifying bounding boxes. According to our statistics, the maximum probability occurs when height-width ratio is 0.72. Hence, this ratio is used as a reference when modifying proposed bounding boxes.

4.1.3 Reducing Overlapped Regions

Due to the watershed algorithm of MSER, many regions generated successively are overlapped. As shown in Figure 4.11a, there are six MSER regions generated around a similar area. As a result, the corresponding bounding boxes are closely located, as shown in Figure 4.11b. Therefore, besides the steps of filtering out undesirable regions by geometry property, we can further reduce the amount of ROIs by cutting down overlapping regions.

(a) (b)

Figure 4.11: Overlapping MSER regions. (a) Six overlapping MSER regions fitted with ellipses. (b) The fitted bounding rectangles to the overlapping MSER.

As mentioned in section 4.1, decreasing the value of maximum intensity variation can result in less regions but may miss some important ones. In this thesis, we combine

overlapping regions to reduce the total number of bounding boxes and keep the variational areas. Through the four-coordinates tuples of each ROI, we can derive the top-left corner coordinates, width and height—which represent the position and size of the bounding box. Regarding the tuples as point coordinate in 4-dimension space, the bounding boxes can be clustered through distance between points. Consequently, using the resulted centroid to substitute the whole cluster contributes to reduce the number of 4-d points. In terms of ROIs, the combination is implemented by clustering all the ROIs and then using the average rectangle of each ROI cluster to represent the group of duplicate regions. As shown in Figure 4.12, using clustering method [2], the closely neighboring bounding boxes in Figure 4.11 are finally combined into one rectangle.

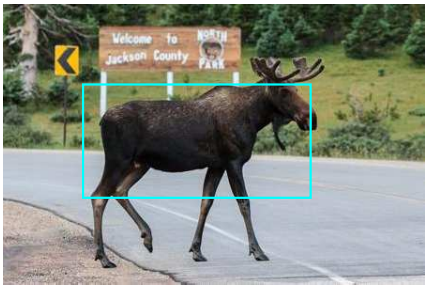


Figure 4.12: Clustered bounding box for 8 overlapping MSER bboxes

Two clustering methods are considered in this thesis: the simple k-means and linkage clustering algorithms. k-means is easy to implement and it needs a prior k value before executing the clustering. However, it is unreasonable to fix the value of k for different training images and testing images. Because the number of MSER regions that could be detected from the input is varying from tens to hundreds. A compromising method is to make the value k adaptive to the number of MSER bounding boxes. For instance, after obtaining the MSER regions from the raw input, the amount of the regions is known to be m. A smaller value k used to cluster all regions should be proportional to m. Dividing m by a constant larger than 1 results to k. The constant used in the paper is 1.5. The larger the constant, the less regions we obtain and the more likely to miss regions.

Hierarchical clustering is a method which seeks to build hierarchy of clusters. There are two types of strategies for hierarchical clustering: agglomerative and divisive. Agglomerative approach begins with each observation as a cluster and then merges pairs of clusters as moving upward along the hierarchy. Divisive approach behaves the opposite: all observations start in one cluster and when we move downward the hierarchy, original cluster splits recursively. In our work, the agglomerative clustering method single-linkage clustering is used for hierarchical clustering.

Single-linkage method is also known as nearest neighbor clustering. It defines the distance between two clusters as the smallest distance between objects in the two clusters.

[2] k-means is used for this example.

From equation 4.6, A and B are two clusters that we want to decide whether to merge them. In the end, we will build a cluster tree based on the minimum cluster distance.

$$d(A, B) = \min d(x, y) : x \in A, y \in B \tag{4.6}$$

Single-linkage does not need a prior value k, the number of clusters. A threshold distance c is used to cut the tree into clusters. In this process, distance from different clusters is compared to c. One cluster is created by a node and all its sub-nodes when all these nodes have inconsistent value less than c. The problem is still existing that we cannot assign a fixed distance value because of different image data. Considering that the distance should be smaller than the shortest boarder of all the bounding boxes, we employ an adaptive method by making $c = min(border)/3$. The shorter the distance, the more regions we achieve.

We evaluate the clustered regions from both k-means and single-linkage by recall/IoU-Threshold curve. By comparing the recall rate of the clustering rectangles through these two methods to the original bounding boxes computed from EL, i.e ellipses, we get three curves shown in Figure 4.13. The recall IoU threshold curve will be discussed in later section. All curves perform to expectation that the recall rate decrease along the rising of the intersection overlap ratio threshold.

The first phenomenon from this Figure is that the recall/IoU-Threshold curves of two clustering methods perform similarly to the recall rate without clustering. It implies that although the number of regions are reduced by clustering, the expected ROIs still exists. That is to say, clustering does not affect the performance of region proposal. The other conclusion is that two clustering methods do not differ in the performance of region proposal. The reason is we are taking adaptive parameters for both clustering methods. Based on such result, clustering methods are not distinguished in analysis of later sections.

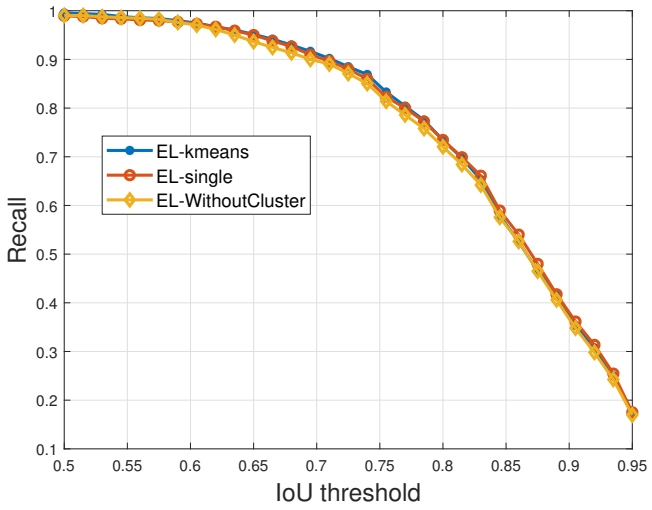


Figure 4.13: Recall IoU curve comparison for two clustering methods

(a) (b)

Figure 4.14: Modification and enlarging of MSER box. (a) Extend the shorter by fixed rate. (b) MSER box is enlarged along both width and height by 1.1 to 1.5, stride=0.1.

4.1.4 Enlarging and Modifying MSER regions

After two steps of reducing the number of ROIs, we have eliminated some unhelpful regions. Another problem is that the MSER bounding boxes cannot cover the entire animal area. Therefore, we need to take some measures to obtain bounding boxes that could cover the entire target to avoid mis-classification. Using the minimum enclosing rectangle of the region's pixel list, with the result in Figure 4.4, the majority part of the animal could be covered. Using the rectangle of the fitting ellipse, with the example shown in Figure 4.6b, as the bounding rectangle cannot cover the ellipse, consequently it cannot cover the pixels of the MSER region. Thus, we need to enlarge the bounding boxes to cover the animal area more precisely, especially for the second method. There are several modifying and enlarging methods that could be used.

(1) MSER boxes are modified if the ratio is scored low by the rate model. Specifically, for boxes whose ratio of long border over short border is taking a low score, the shorter border is enlarged to make an optimal ratio. The optimal ratio is selected as 0.8. In other word, we extend the axis along which the border length is shorter than the other axis. As shown in Figure 4.14a, the original box in blue is modified to the purple one. As the height of the original MSER box is much shorter, we extend the height to a larger value. For this example, the height of the blue box is extended to 0.8 of the width.

(2) MSER boxes are enlarged along both x and y axis. The simplest enlarging method is to extend the width and height of the original rectangle by multiplying some factors. The resulted bounding box of Figure 4.14a still does not cover the whole animal. It is then extended in both horizontal and vertical directions by factors $1.1, 1.2, 1.3, 1.4, 1.5$ as shown in Figure 4.14b. The enlarged rectangles can cover the moose region better.

(3) MSER boxes are enlarged along one $x - axis$ or $y - axis$ axis. Based on the simplest enlarging results, the enlarged bounding boxes may not correctly represent the ratio of the animal if the animal is taller or longer than the MSER box, especially when the

(a) (b)

Figure 4.15: Extend only width or height of the MSER box. (a) Extend only height of the MSER box. (b) Extend only width of the MSER box. Factors are 1.1 to 1.5, stride=0.1.

(a) (b)

Figure 4.16: Extend more width or height of the MSER box. (a) Extend more height of the MSER box. (b) Extend more width of the MSER box.

animal is hopping or running. Therefore, we need to modify the multipliers adaptively for width and height. We enlarge only width or height, as shown in Figure 4.15. Enlarging along only one axis may result in proper bounding box. In this example, Figure 4.15a achieves better bounding boxes as the moose in this image has horns and legs along the height.

(4) MSER boxes are enlarged by different factors along $x - axis$ or $y - axis$. To make up some missing pixels in different axis, we enlarge the bounding by multiplying varying factors along x and y axis. As shown in Figure 4.16a and 4.16b, the original purple boxes are enlarged only along height and width respectively.

(5) MSER boxes can also be enlarged along one directed axis only. As the feet of animals tend to be missed in MSER regions, we can extend the height only to the bottom direction. Similarly, the tails part of animals are often missed. We can include extra part along width from only one direction.

(6) MSER boxes can be enlarged by two intersecting borders. We can enlarge the MSER bounding box from directions of bottom and right. With two directed axis combined, corner or enlarge from top and right, from top and left, from bottom and left can be performed.

In this thesis, MSER boxes whose ratio obtains a lower score by Equation 4.4 are modified first. We select multiple enlarging approaches to make the bounding box more robust. The proposed ROIs are augmented by adding the enlarged ROIs. As more detailed enlarging method implies more computation cost, we are not using all the enlarging methods mentioned but select only two of them.

In order to show the benefit of the modification and enlarging, we draw the recall IoU threshold curve in Figure 4.17 where both PL and EL methods are analyzed. In both figures, the curve representing the original bounding boxes is lower than that of the modified one which is lower than that of the modified and enlarged one. Modifying the original bounding boxes and then enlarging them benefit in raising the recall rate of the proposed regions.

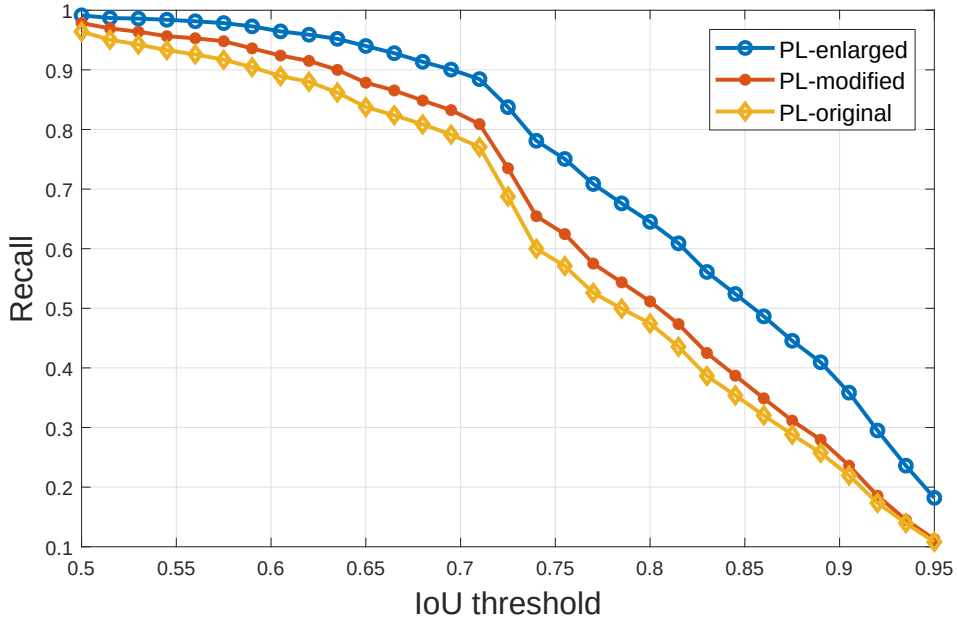
4.1.5 Summary

MSER detection performs the first step in our animal detection framework. We extract the bounding boxes through two methods. The first one is PL MSER which encloses the MSER region represented by the pixel list. The other is EL MSER one which fits the region to an ellipse and then fits to a rectangle. PL MSER is intuitive and EL MSER is able to bound the animal by its centroid. The resulted bounding boxes are then filtered and scored by the ratio distribution which is estimated by the dataset. We then cluster the bounding boxes to reduce the overlapping of the MSER regions. At last, we modify the ratios of bounding boxes and augment the bounding boxes by enlarging them to better cover the animal.

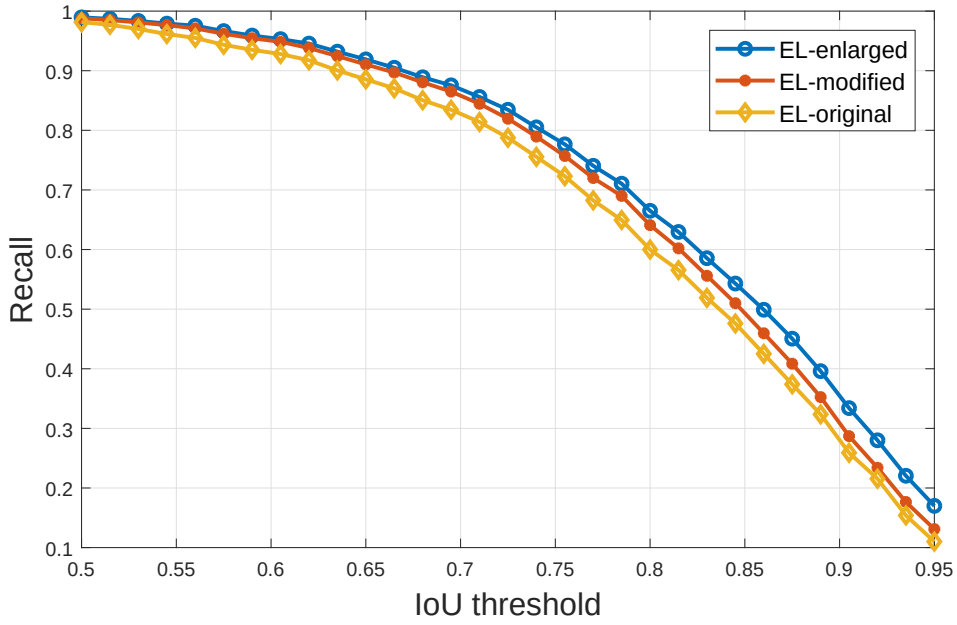
4.2 R-CNN with MSER

We propose a new R-CNN framework by applying MSER region proposal, as shown in Figure 4.18. Using MSER as region proposal instead of Selective Search or Edge Boxes, fewer regions are generated while keeping the coverage against the ground-truth bounding boxes. The novel R-CNN detection system consists of 4 steps: MSER detection, CNN feature extraction, SVM classification and bounding box regression. We begin with generating candidate ROIs through MSER detection and preprocessing. The generated bounding regions are then warped and resized to match the following CNN architecture. The convolutional layers of CNN extract features of fixed size from the input regions. CNN features of the same size are then fed into the SVM classifier for classification. An extra step of bounding box regressor is used to refine the position and size of the bounding box.

Among these four steps, parameters of the SVM classifier and bounding box regressor needs to be learned from training. At test time, the classifier parameters contribute to deciding whether the regions are animals or not. We select the strongest bounding boxes among the positive results according to the rule of Non-Maximum Suppression (NMS). The regressor parameters are then used to refine the bounding boxes.



(a)



(b)

Figure 4.17: Recall/IoU-Threshold curve when modifying and enlarging MSER bounding boxes. (a) Comparison among original, modified and enlarged regions using PL MSER. (b) Comparison among original, modified and enlarged regions using EL MSER.

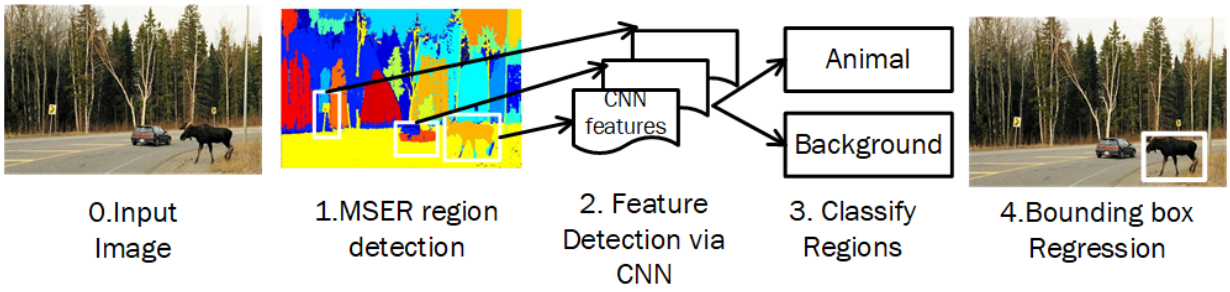


Figure 4.18: MSER applied in R-CNN Framework.

4.2.1 CNN Feature Extraction

The resulted regions from the input image are warped and scaled to a fixed size of 227×227 to fit the CNN layer. The warped ROIs are shown in Figure 4.19. We can see that all bounding boxes are scaled to the same size so that it matches the input layer of the CNN structure. However, the extra scaling step have the proposed region deformed. The deformation brings error to feature recognition, which is why Fast R-CNN and Faster R-CNN try to get rid of this step.

Figure 4.19: Warped ROIs of animals.

CNN structure for feature extraction

We the pre-trained AlexNet to extract CNN features, as shown in Figure 4.20. The structure of AlexNet for image classification has been studied in section 3.1.2. As shown in Figure 4.20a, the pre-trained network contains 8 layers to classify 1,000 categories of the ImageNet dataset. After 5 convolutional layers, fixed size input is needed for the following 3 fully connected layers. The last fully connected layer 'FC8' in AlexNet is essentially a Softmax Classifier to compute probabilities for 1,000 categories.

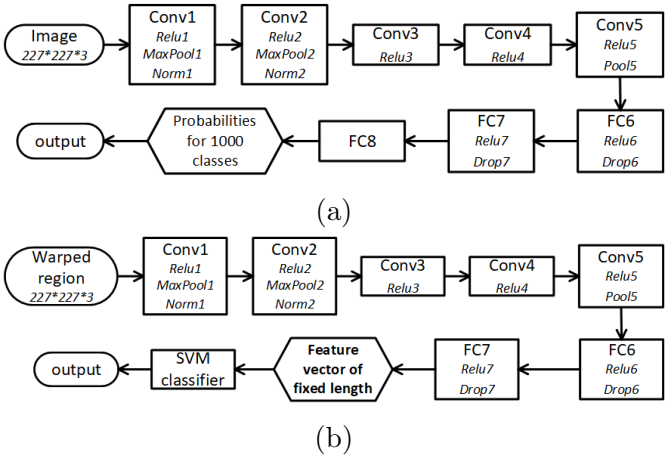


Figure 4.20: AlexNet Structure.(a) AlexNet structure for image classification. (b) AlexNet structure for feature extraction.

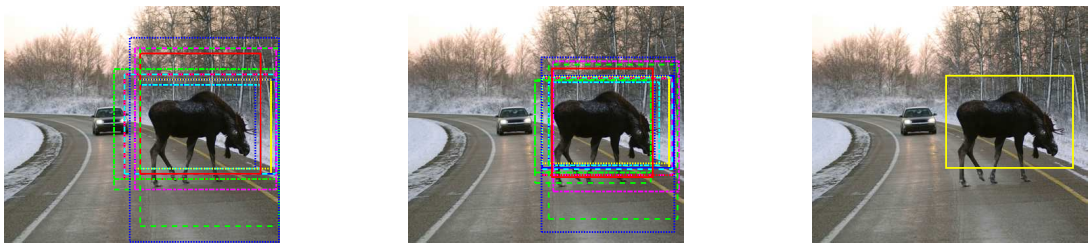
In our framework, we do not train the ImageNet again but use AlexNet as a feature extractor. The structure is modified for the purpose of classifying animals from background. As shown in Figure 4.20b, the input image is actually one ROI or cropped region by the proposed bounding box which has been scaled to a fixed spatial size 227×227. 'FC8' is replaced by a linear Support Vector Machine (SVM) classifier, which is trained by our own dataset. The SVM classifier accepts a feature vector of size 1×4096 from the 'FC7' layer and labels the feature as 'Animal' or 'Background' by the calculated class scores. To represent a more intuitive result, we can change the hinge loss function in SVM to cross-entropy loss to calculate the confidence of each category. Switching between softmax and SVM is simple and useful for classification tasks [113].

4.2.2 Bounding Box Post-process

After the step of classification, we obtain the confidence of animals' existence in each region. As the numerous regions cropped from the image are overlapped, we then need to suppress some non-maximal regions and transform the bounding boxes to represent the animal better.

For regions which are labeled positive, a bounding box regressor is trained to transform the proposed regions as close to the ground truth bounding box as possible. The input of

the regressor is the feature vector of fixed length (4,096-d) and the output is the learned parameters to transform the proposed bounding box. In test time, the first operation to post-process the bounding boxes is to fine-tune them by the pre-trained offset parameters, together with the labels and classification scores. As an example shown in Figure 4.21a, a dozen of bounding boxes are proposed and each of them is transformed accordingly to Figure 4.21b.



(a) (b) (c)

Figure 4.21: Post-processing of bounding box. (a) Extracted MSER bounding boxes. (b) Regressed bounding boxes. (c) The strongest bounding box selected from (b).

The bounding box regressor is followed by non-maximum suppression. After the regression, we still get a lot of overlapping bounding boxes as illustrated in Figure 4.21b. Therefore, we need suppress the bboxes by removing overlapping regions. The method is based on overlap ratio stated in section 3.4.1. Instead of applying IoU to measure the detection, we take the measurement of IoM (Intersection over Minimum) to limit the number of bboxes. If two bounding boxes intersect and the intersection area is greater than a percentage of the minimum area (such as 50%), we select the stronger bbox with larger probability and get rid of the weaker one. After suppression, the detection result becomes clear as shown in Figure 4.21c.

4.2.3 Summary

In this section, we apply MSER in R-CNN framework. The last layer of the pre-trained network is fine-tuned in adaptive to the task of animal detection. After generating bounding boxes using MSER, we input the cropped regions into the CNN network for feature extraction. The resulted scores contributes to mark the positive bounding boxes, which are then transformed by bounding box regressor.

4.3 Fast R-CNN with MSER

The framework of R-CNN with MSER in section 4.2 describes the idea of applying MSER in object detection. However, as the proposed regions are warped in R-CNN, the aspect ratio of the animal is changed consequently. The extracted feature cannot represent the animals

exactly representation. Additionally, the framework needs to extract CNN features from all proposed regions. No matter how many ROIs are generated using MSER algorithm, the execution of CNN is not shared. Thus the computation is expensive. These problems motivate us to apply MSER in a better version of R-CNN—fast R-CNN.

As shown in Figure 4.22, we propose a new framework of fast R-CNN using MSER detection as region proposal method. Each generated MSER region is represented by a tuple (x, y, h, w), where x and y denote coordinates of the region's left-top corner, h and w denote the height and width of the bounding boxes. Feature maps of arbitrary-sized input image are obtained from the convolutional layers of the CNN network. ROIs represented by (x, y, h, w) are then reflected to fields in the feature map. The ROI pooling layer then extracts a feature vector of fixed length from each field. The ROI pooling layer is followed by fully connected layers. In the end of the framework, two sibling output layers manages the result of fully connected layers. The Softmax classifier produces the probability at which the ROI contains animals. The bounding box regressor transforms the positive bounding box to a more accurate location.

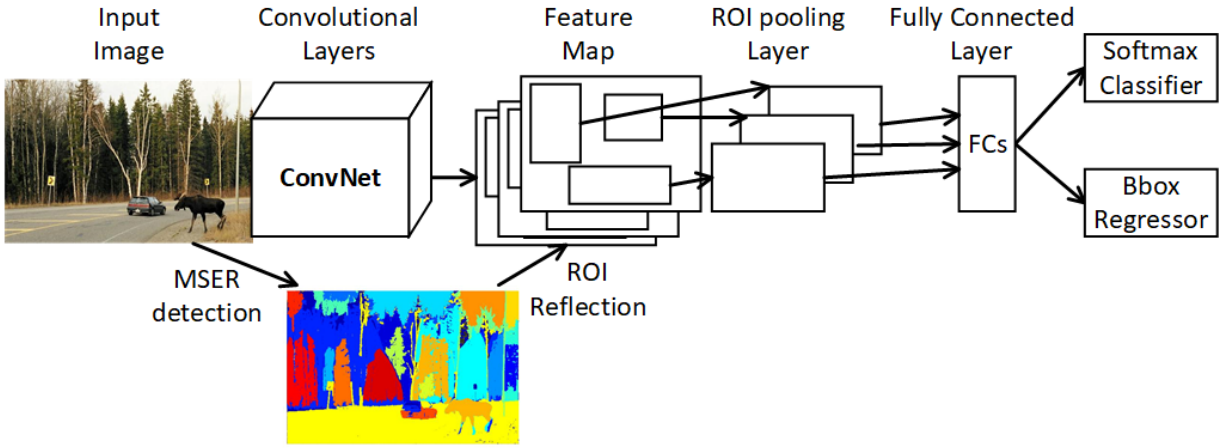


Figure 4.22: MSER applied in fast R-CNN Framework.

As stated in 3.2.2, CNN computation is shared among all proposed MSER regions. Due to the use of ROI pooling layer, a fixed-length feature vector is extracted from each MSER region of arbitrary size. For the example of ZFNet, ROI pooling layer is placed between the block of Conv5+ReLu5+Pool5 and the block of Fc6+ReLu6+Drop6. The ROI max pooling layer is configured with windows of size 8×8.

4.4 Dataset

The dataset used in this thesis is extended from [130] where the animal images dataset is obtained by capturing videos using the ratio 1 per 4 continuous frames to avoid repeated sampling. The dataset is huge but only consists of deer and moose, and most postures of

the animals are heading to left or heading to right. Samples of animals in different lighting condition, such as night, are lacked.

To enhance the diversity and representativeness, this dataset plus are created covering the following improvement.

(1) More species of animals. Cervidae and Equidae including young and adult animals are contained, like moose, deer, elk, caribou, horse, mustang. Where the horse category is obtained from the dataset of PASCAL VOC 2012.

(2) More postures of animals. Not limited to strolling observed from a lateral view, this enhanced dataset includes more animal actions, like sitting, running, howling, turning around, heading forward and heading backward,

(3) More scenes are considered. Except the frames captured from videos, images are also obtained from various website, such as the site of Parc Omega.

(4) Animal images are not cropped out of the background but negative images are not obtained by extra effort. The negative image can be achieved from the background area in program.

(5) Multiple animals, no matter whether they belong to the same species, in one image are included in the dataset.

In summary, the dataset is an advanced version of that used in [130]. As shown in Figure 4.23, different categories are included in the set. It is difficult to tell the difference from elk to deer. The detection aims to find the animals in the image regardless of their category.

(a) Moose

(b) Elk

(c) Deer

(d) Horse

Figure 4.23: Dataset of different categories.

Chapter 5

Results and Analysis

As stated in chapter 4, we implement the animal detection system by applying MSER region proposals in R-CNN and fast R-CNN framework. Involving multiple technique steps, the detection system may achieve different results due to various choice of options. In this chapter, we analyze the parameters and options that may affect the result and get a summarized conclusion of the affecting parameters. Using the optimal options based on the conclusion, we compare the results of our region proposal method to edge box and RPN. We then compare the result of our detection method to the state-of-the-art faster R-CNN in multi-folds.

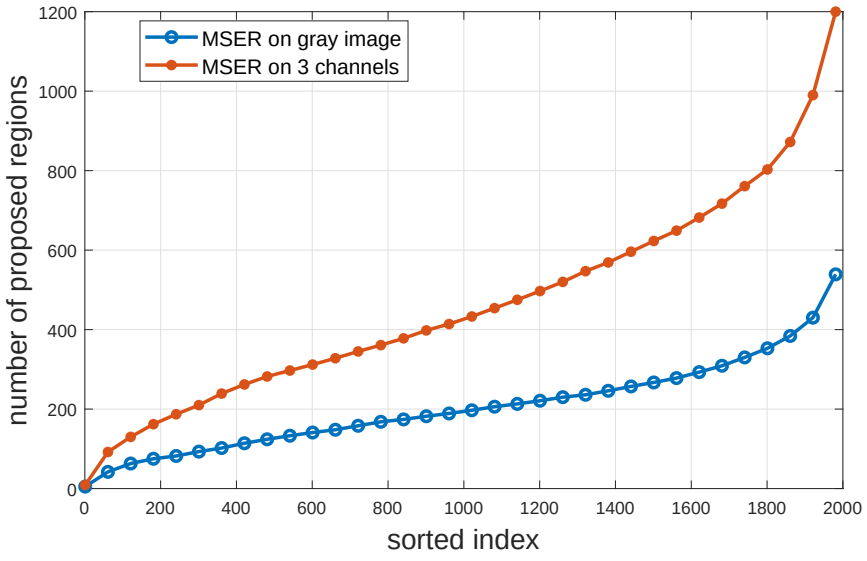
5.1 Contributing Parameters

The steps of MSER detection and CNN involves many parameters and options which may affect the final result. This section is to analyze these parameters and options to find a optimal combination for animal detection.

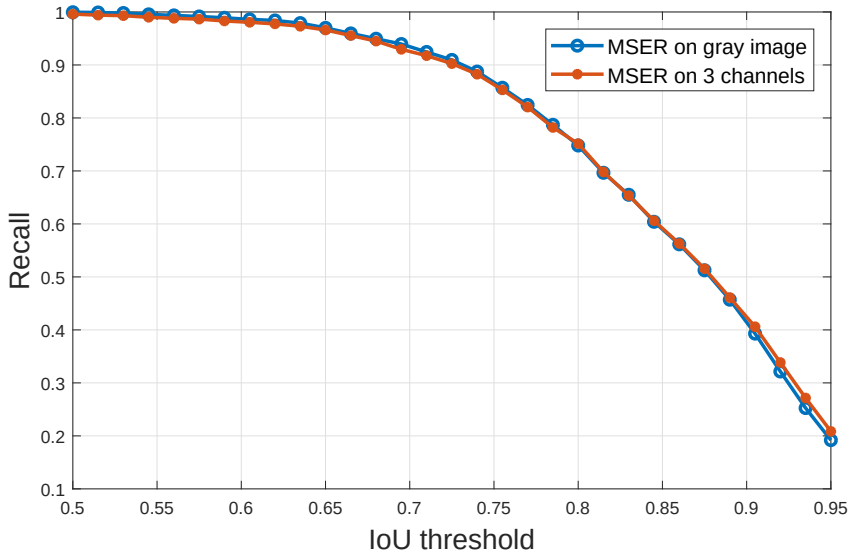
5.1.1 Gray MSER or color MSER

MSER algorithm could be applied to both gray image with single channel and color image with three channels. To measure the performance of processing gray and color image, we select 2,000 images randomly from the dataset and generate candidate regions from them using the same PL MSER method. The result shows that the amount of regions proposed from three channels is about 2.3 times of that from gray image in statistics. As shown in Figure 5.1a, the number of MSER regions generated from all three channels is much larger than that from grayscale image. However, the increasing of amount of proposals does not equally increase the coverage of the proposals. As shown in 5.1b, region proposals from either source achieves very similar recall rate over IoU threshold.

From the result of Figure 5.1, although MSER applied to three channels produces more candidate regions, it does not outperform gray MSER in terms of recall. Therefore, we



(a)



(b)

Figure 5.1: Comparison of gray MSER and color MSER.(a) Number of proposed MSER regions extracted from gray image or color image with 3 channels. (b) Recall over IoU threshold curve of MSER regions extracted from gray image or color image with 3 channels.

only extract MSER regions from the grayscale image of the input dataset to save time and memory.

5.1.2 R-CNN or fast R-CNN

As stated in section 4.2 and 4.3, MSER region proposal can be applied to both R-CNN and fast R-CNN framework. In theory, fast R-CNN saves computation cost by sharing convolutional layers and feature maps. We then implement both frameworks with MSER region proposal using ZFNet.

PL MSER is applied to both fast R-CNN and R-CNN, with the detection results shown in Figure 5.2. The PR curve of fast R-CNN is higher and longer than the curve of R-CNN. The higher PR curve denotes higher precision of fast R-CNN. Besides, the average precision along all ranges of recall for the two methods are 0.71 and 0.45 respectively. The longer curve denotes better coverage of all positive regions. Higher average precision is achieved using fast R-CNN. The miss rate in Figure 5.2b, where the miss rate curve of fast R-CNN is lower than R-CNN, draws the same conclusion that applying MSER region proposal in fast R-CNN performs better than in R-CNN on both precision and recall rate.

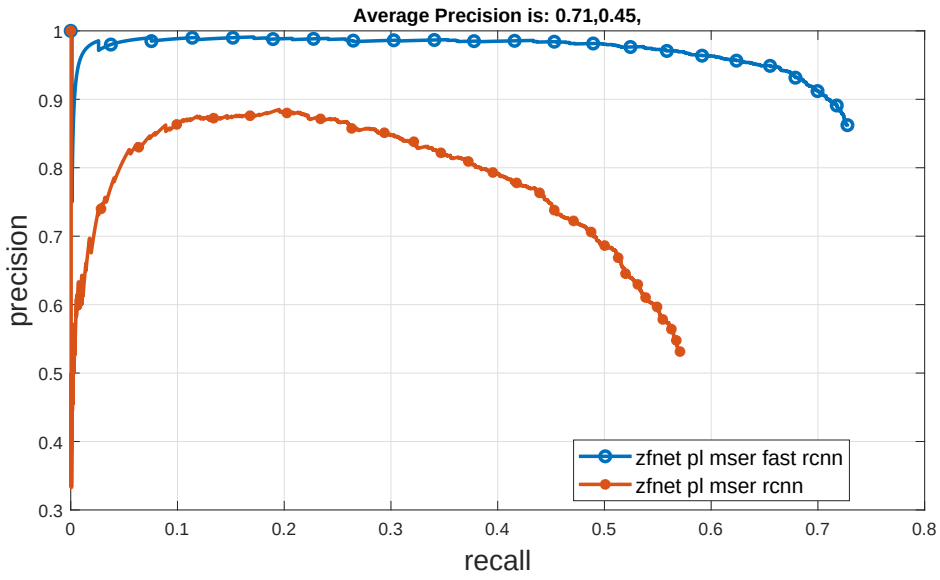
5.1.3 The depth of CNN

The depth of CNN network has a compact on the recognition accuracy, thus affecting the accuracy of detection [110]. We apply PL MSER region proposal in fast R-CNN using three networks. The first one has only four layers, the second one has totally 5 layers and the third one is ZF net. ZF network is implemented with the parameters shown in section 3.1.3. The other two networks are configured as in table 5.1. The 4-layer network is composed of only two convolutional layers and two fully-connected layers. The 5-layer network consists of three convolutional layers and two fully-connected layers.

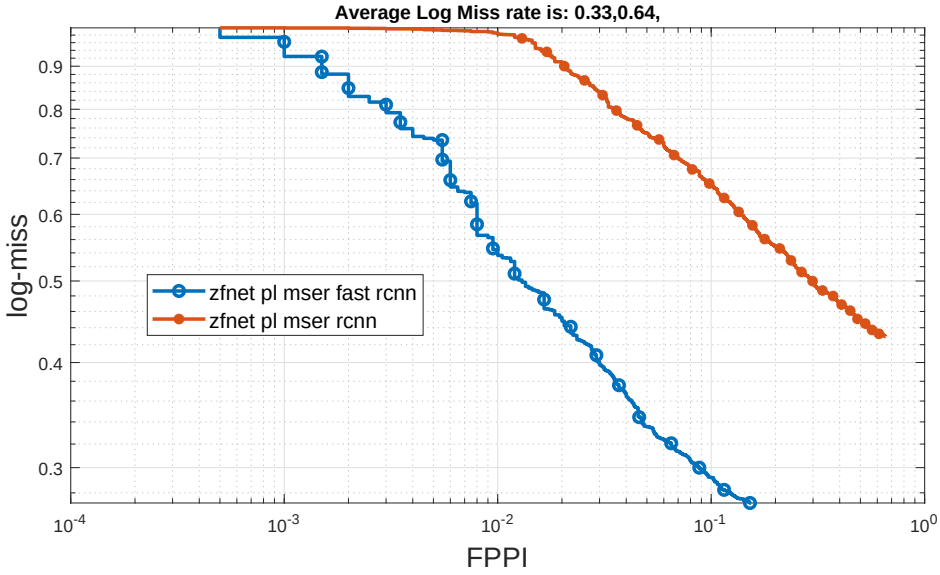
Table 5.1: Parameters of 4-layer and 5-layer network

4-layer Net	**5-layer Net**
input image($227 \times 227 \times 3$)	input image($227 \times 227 \times 3$)
Conv1: 32, $\mathbf{5 \times 5, stride = 2, pad = 0}$ ReLu1	Conv1: 32, $\mathbf{5 \times 5, stride = 2, pad = 0}$ ReLu1
Conv2: 32, $\mathbf{3 \times 3, stride = 2, pad = 1}$ ReLu2	Conv2: 32, $\mathbf{3 \times 3, stride = 2, pad = 1}$ ReLu2
Pool: max, stride=2, $\mathbf{pad = 0}$	Pool: max, stride=2, $\mathbf{pad = 0}$
	Conv3: 64, $\mathbf{3 \times 3, stride = 1, pad = 1}$ ReLu3
	Pool: max, stride=2, $\mathbf{pad = 0}$
Fc3: 64 ReLu3	Fc4: 64 ReLu4
Fc4: num_cls	Fc5: num_cls

Applying PL MSER in the three networks, the result of precision recall and log miss rate are shown in Figure 5.3. The average precision of the three networks are 0.54, 0.60 and 0.67 respectively. The best performance is achieved in ZFNet. The performance comparison



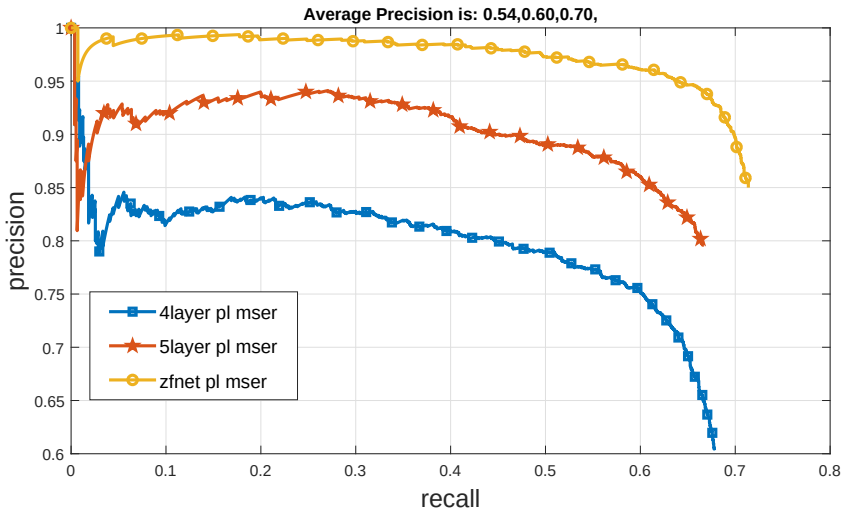
(a)



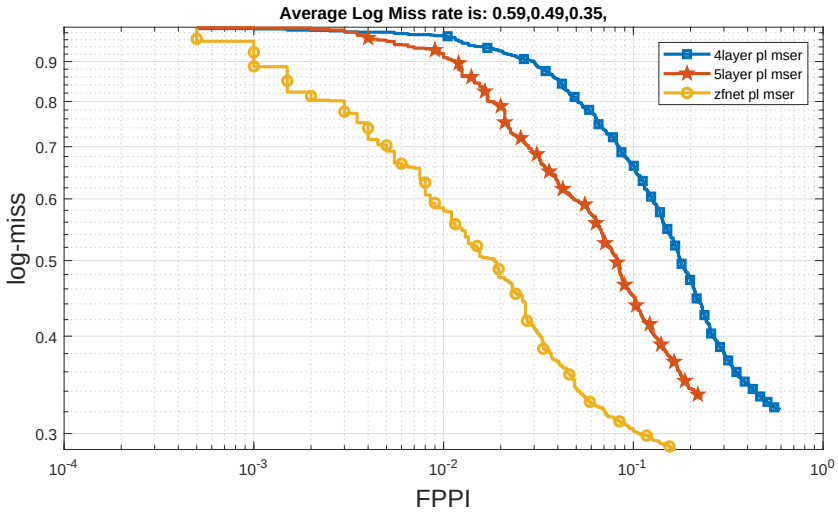
(b)

Figure 5.2: Comparison of R-CNN and fast R-CNN.(a) PR curve comparison. (b) Log miss rate comparison.

from the curve trend, as well as from the logmiss curve and value, are consistent to the conclusion. As 8-layer ZFNet performs better than the simple 4-layer and 5-layer network, we implement the animal detection system by applying MSER on fast R-CNN based on ZFNet.



(a)



(b)

Figure 5.3: Compact of network depth.(a) PR curve comparison. (b) Log miss rate comparison.

5.2 Analysis on MSER Region Proposal

To evaluate the region proposal algorithm, we compare the proposed method of MSER using pixel list (PL MSER) and ellipses (EL MSER) with the method used in fast R-CNN and faster R-CNN, which are edge boxes and RPN respectively. In this section, we evaluate the quality of detection proposals based on the number of proposed regions and the recall of the ground truth annotations.

As shown in Figure 5.4a, the number of generated regions are sorted and plotted for different region proposal methods. The method of Edge Box proposes the largest number of regions while RPN proposed the least. Curves of PL MSER and EL MSER fall in

between. Specifically, EL MSER proposes more regions than PL MSER. The reason is EL MSER takes more enlarging measures to augment the proposals. There is a particular case for RPN that a large part of RPN line stays for zero. Such phenomenon can be explained by that RPN proposes regions based on the convolutional features and misses candidate regions due to false recognition. As far as we know, a greater number of proposals does not indicate higher recall to the expected regions. Less number of proposals would reduce the cost to compute features. Therefore, the PL MSER and EL MSER region proposal methods generate candidate regions more efficiently, compared to Edge Box used in original fast R-CNN.

In terms of recall rate, we compare among the four proposal methods by computing the curves of Recall/IoU threshold as shown in Figure 5.4. When the IoU threshold is in the period $[0.5, 0.6]$, all curves of the four proposal methods reach up to 100%. But when the threshold is raised higher, recall rate of PL MSER and EL MSER keeps higher than edge boxes and RPN methods. Observing the curves of EL MSER and PL MSER, PL MSER and EL MSER performs similarly although EL MSER proposes more regions.

Above all, the proposed methods generate fewer candidate regions and achieves a higher recall than edge boxes. Compared to RPN, EM MSER and PL MSER produce more regions but the recall rate is higher. Although the proposed regions are not necessarily detected correctly by the system, a higher recall contributes to reduce the missing rate of detection.

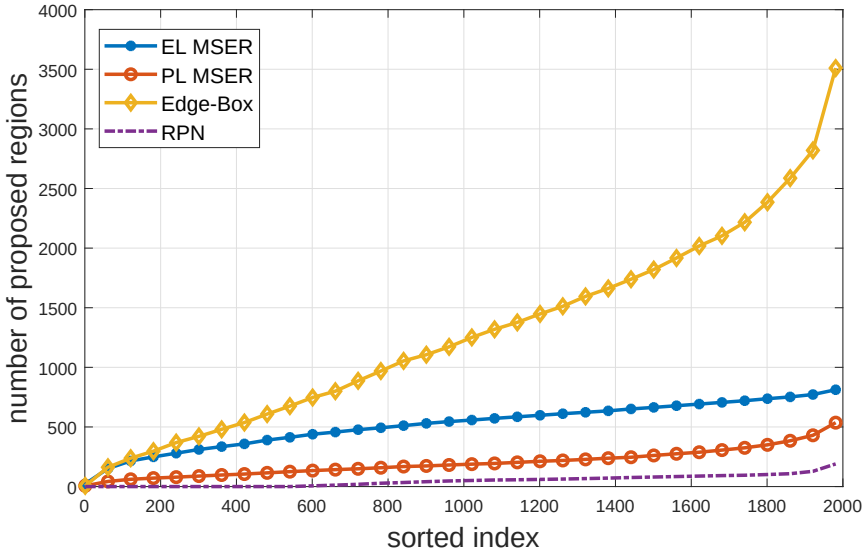
5.3 Detection Result

To evaluate the detection method, we compare the detection results of the proposed algorithm and the classic faster R-CNN [1]. In terms of effectiveness, we separate the detection results into several cases as shown in Figure 5.5. The green bounding boxes in these figures denote the expected ones which are marked manually for the dataset. The yellow bounding boxes are the detection result from faster R-CNN while red and blue bounding boxes are detection results from EL MSER and PL MSER respectively.

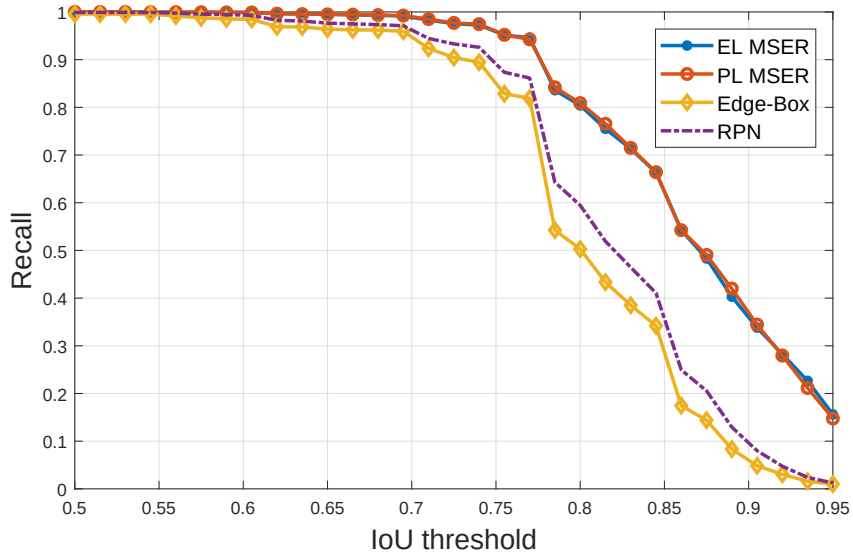
In these results, the proposed algorithm performs better than faster R-CNN in three cases. The first case is that faster R-CNN detects part of the animal while the proposed method detects the whole animal. This is because RPN in faster R-CNN proposes regions in the way of sliding window. Although the proposed ROIs are processed and optimized by the bounding box regressor, the manner of sliding window does not generate regions based on objectness. It is likely to miss the object's actual region. A similar reason leads to the second case that multiple parts of the animal are detected by faster R-CNN while the proposed can detect correct numbers of whole animal. In the third case, faster R-CNN fails to locate the animal in tested image while the proposed detection method succeeds.

Besides the above cases when the proposed outperforms faster R-CNN, there are cases when the proposed does not perform better. As shown in 5.6, faster R-CNN detects tighter or equally accurate bounding boxes compared to the proposed. To evaluate the proposed

[1] ZF net are used in faster R-CNN and PL MSER, EL MSER algorithms.



(a)



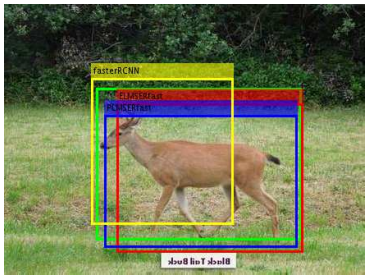
(b)

Figure 5.4: Comparison among region proposal methods. (a) Comparison on number of proposed regions. (b) Comparison on Recall/IoU-Threshold.

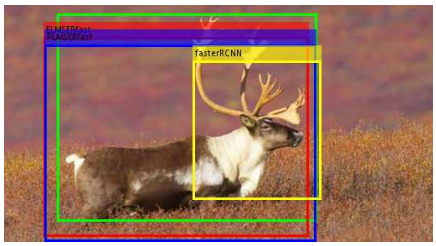
algorithms precisely, we need the measurements introduced in 3.4 to analyze the detection results quantitatively.

5.3.1 Analysis of PR curve

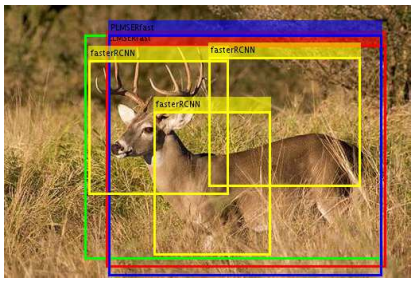
To evaluate the detection result more comprehensively, we take experiments on our dataset using the region proposal algorithm PL MSER and EL MSER applied in both fast R-CNN



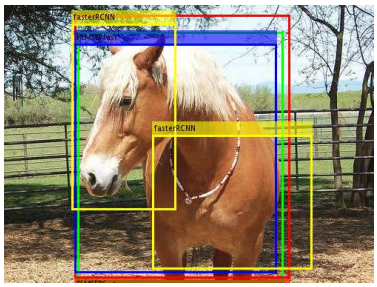
(a)



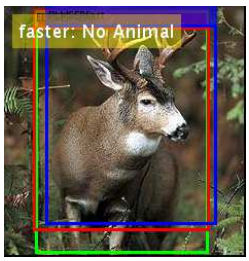
(b)



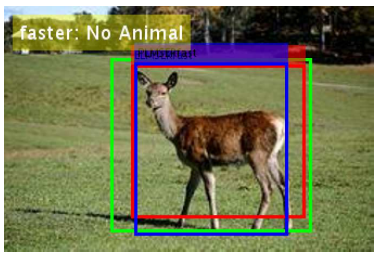
(c)



(d)



(e)

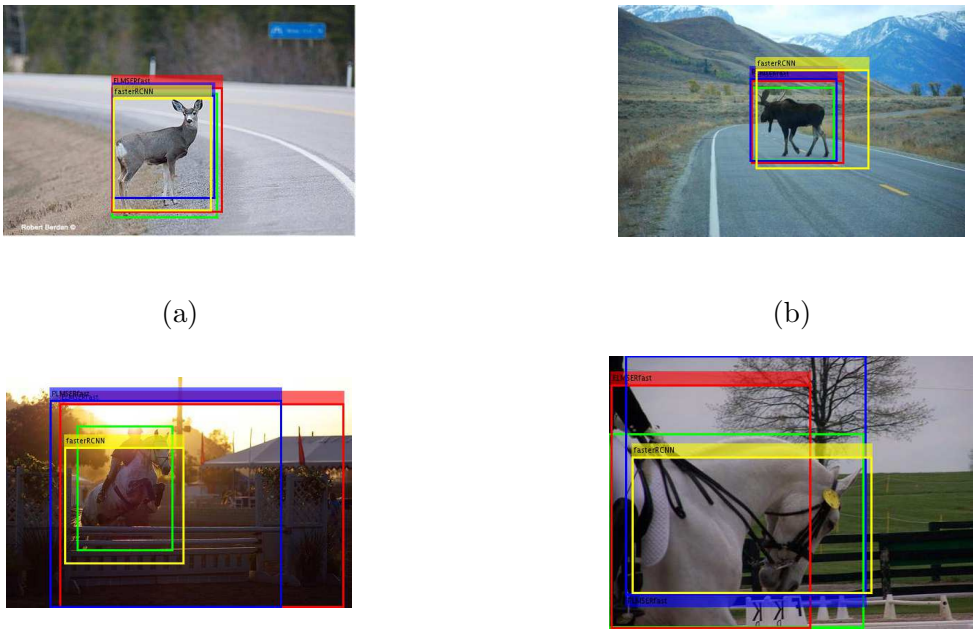


(f)

Figure 5.5: Better detection result examples. (a)(b) The proposed detects the whole animal while faster R-CNN detects part of the animal. (c)(d) The proposed detects correct number of animals while faster R-CNN detects more than expected. (e)(f) The proposed detects the animals while faster R-CNN found no animal.

and R-CNN framework. We employ three networks with different number of layers as CNN structures, as mentioned in section 5.1.3. To compare the proposed detection framework with faster R-CNN [101] and fast R-CNN, we evaluate the detection results by PR curve in this section.

As shown in Figure 5.7, PR curves are plotted for seven different detection methods. The title text also displays the average precision in the same order as in the legend. The PR curve tending to top implies a higher detection precision, while the PR curve extending to



(a) (b) (c) (d)

Figure 5.6: Not better detection result examples. (a)(b) All methods detect the animal well. (c)(d) Faster R-CNN detects the animal with a tighter bounding box.

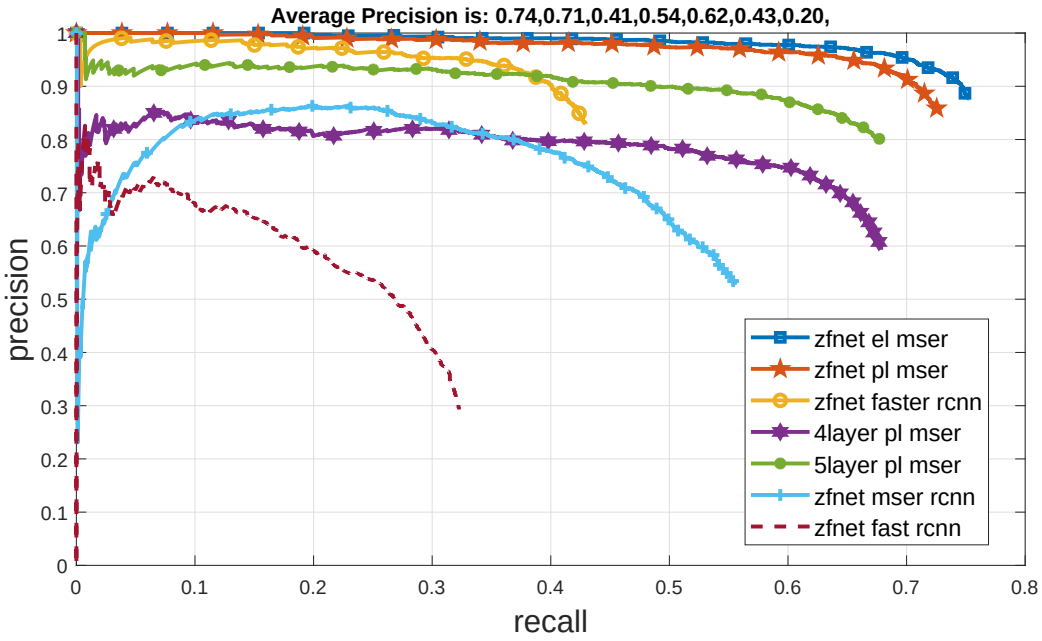


Figure 5.7: Precision-Recall among different algorithms.

the right implies a higher recall. Therefore, the comparison between the PR curves for two proposed methods, 'zfnet el mser' and 'zfnet pl mser', indicates that EL MSER performs

slightly better in detection precision than Pl MSER. With the awareness of that PL MSER outperforms EL MSER in terms of recall IoU threshold curve, we verify the conclusion in [101] that a better recall IoU threshold does not necessarily indicates a better detection. The comparison between PR curves for the proposed detection methods with that of faster R-CNN ('zfnet faster rcnn') suggests that the novel approaches achieves better precision and recall than faster R-CNN.

The evaluation results keep in accord with the conclusion in section 5.1.2 when we compare the PR curves of 'zfnet pl mser' and 'zfnet mser rcnn'. The performance of fast R-CNN is far more better than that of R-CNN with MSER region proposal both applied. We can also draw a consistent conclusion with that in section 5.1.3 by the comparison among the PR curves of 'zfnet pl mser', '4layer pl mser' and '5layer pl mser'. Detection by ZFNet performs better than that by 5-layer network and detection by 5-layer network performs better than by 4-layer network in both precision-recall and average precision.

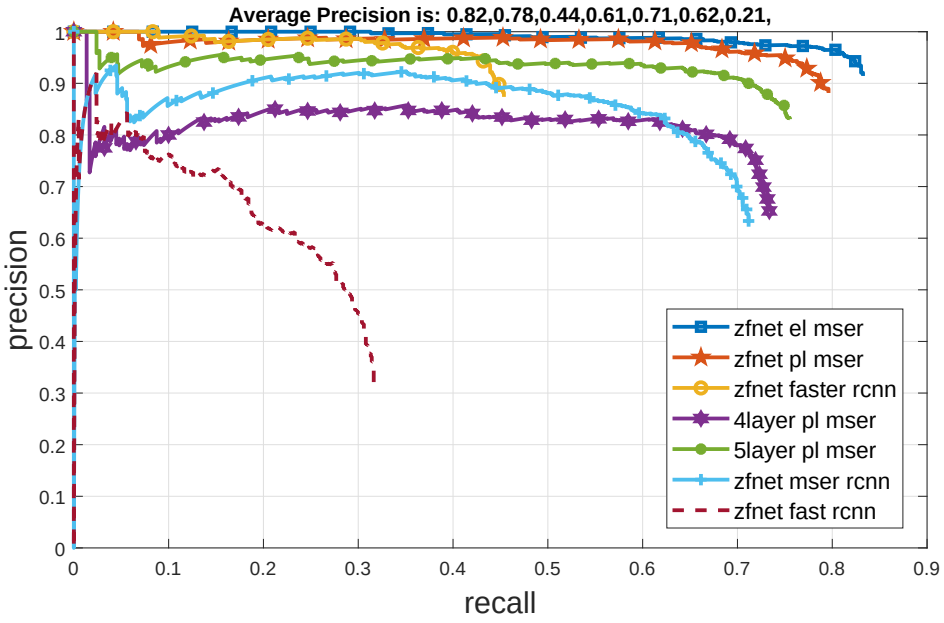


Figure 5.8: Precision-Recall of Moose

To evaluate our proposed detection methods, we also take experiments to the dataset of a single category— moose, deer, elk and horse. Although the proposed detector does not distinguish the category of animals, the analysis of detection on separate category helps us to learn more about the detector. PR curves for the single category of animal are shown separately in Figure 5.8, 5.9, 5.10 and 5.11. The general trends of seven detection methods are similar to that in Figure 5.7. A remarkable part is that the PR curves of moose by the proposed detection methods go far better than faster R-CNN.

As shown in Table 5.2, average precisions of different scenarios are listed. The comparisons in average precisions for these detection methods meet the conclusion in PR curves. The largest average precision occurs in the algorithm of EL MSER applied in fast R-CNN.

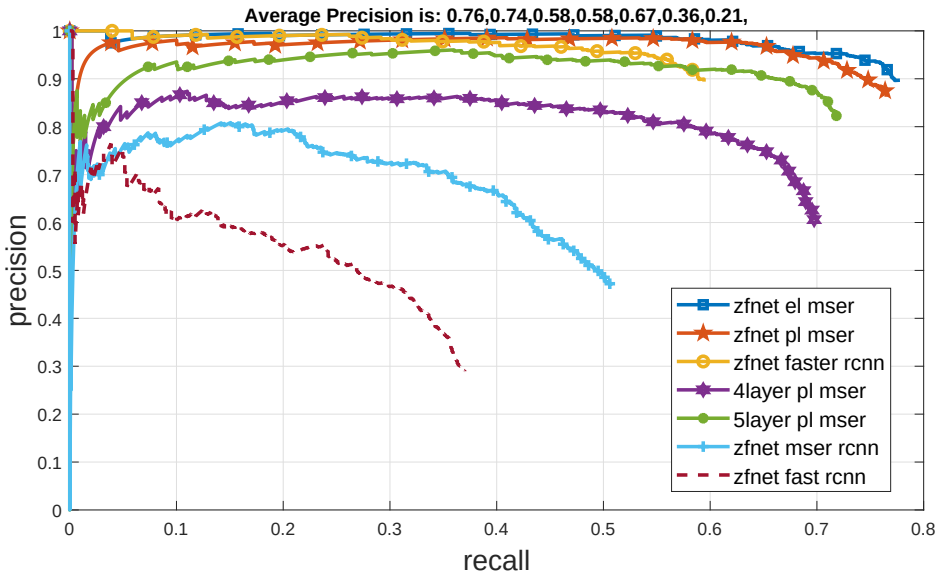


Figure 5.9: Precision-Recall of Moose

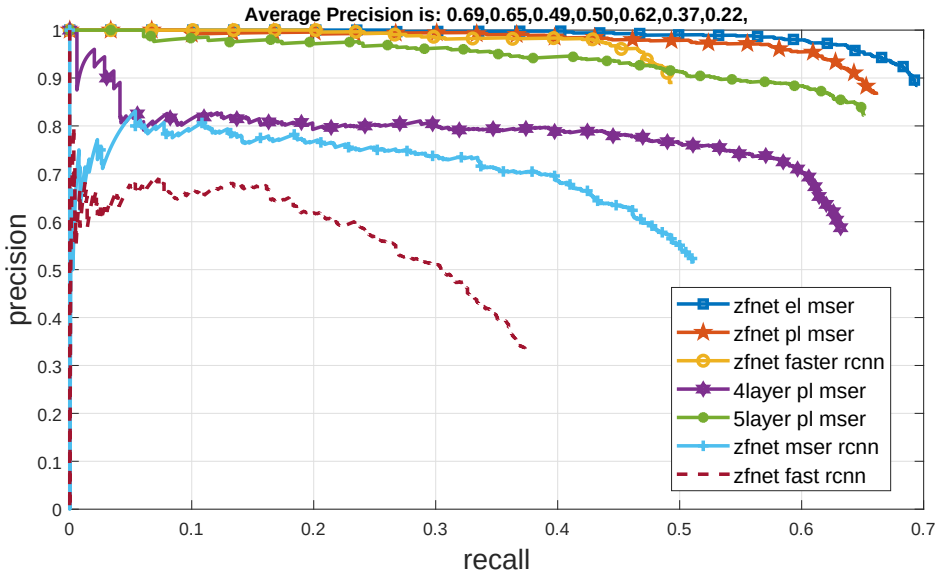


Figure 5.10: Precision-Recall of Elk

5.3.2 Analysis of Miss Rate

Except for the PR curve, we also draw the Miss-rate/FPPI curve in logarithm domain. For this analysis, a lower curve indicates a detection with less miss. We calculate the miss-rate/FPPI curves for the proposed methods and other reference methods using our dataset, as shown in Figure 5.12. The miss rate curve of the proposed methods, 'zfnet el mser' and 'zfnet pl mser', are much lower than that of faster R-CNN, 'zfnet faster rcnn'. The text in Figure 5.12 also provides us the average logarithm miss rate.

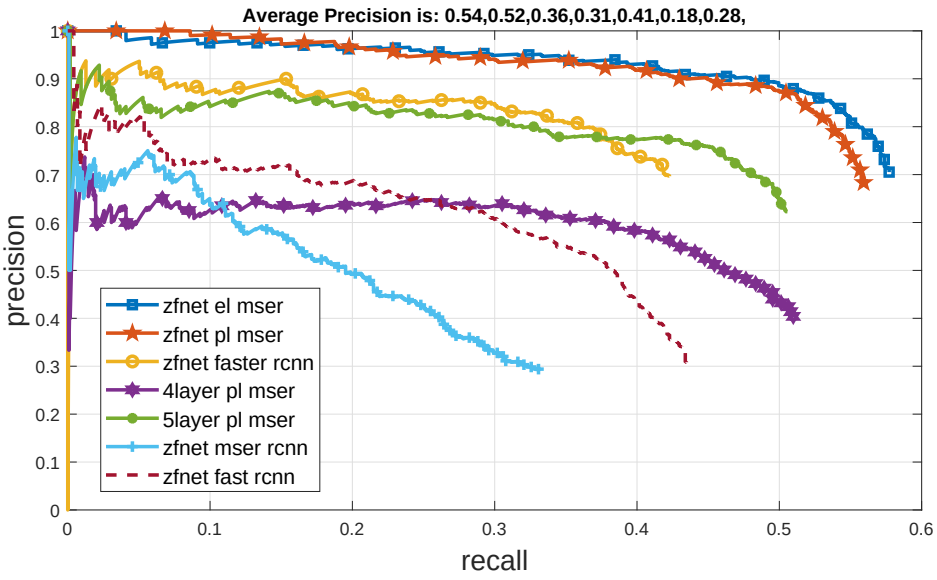


Figure 5.11: Precision-Recall of Horse

Table 5.2: Average Precision result

algorithm	moose	deer	elk	horse	all
ZFnet El MSER fast R-CNN	**0.82**	**0.76**	**0.69**	**0.54**	**0.74**
ZFnet Pl MSER fast R-CNN	0.78	0.74	0.65	0.52	0.71
ZFnet faster R-CNN	0.44	0.58	0.49	0.36	0.41
4-layer PL MSER R-CNN	0.61	0.58	0.50	0.31	0.54
5-layer PL MSER R-CNN	0.71	0.67	0.62	0.41	0.62
ZFNet MSER R-CNN	0.62	0.36	0.37	0.18	0.43
ZFNet fast R-CNN	0.21	0.21	0.22	0.28	0.20

Similar to the analysis of PR curve, we also take the experiments on the single category and plot the miss rate curves separately, as shown in Figure 5.13, 5.14, 5.15 and 5.16. Besides the miss rate curves along recall, we then compute the average logarithm miss rate in values for both single category and all categories, as shown in Table 5.3. The miss rate of both PL MSER and EL MSER region proposal algorithms are indeed smaller than faster R-CNN using same network. The smallest average miss rate occurs from the algorithm of EL MSER.

The figures of 5.12 and Figure 5.13, 5.14, 5.15, 5.16 and Table 5.3 imply a same conclusion as what we draw in section 5.3.1 by an opposite measurement. The curves which stand for the PL MSER and EL MSER lead to smaller miss rate than faster R-CNN and fast R-CNN. The gap between the proposed ones and faster R-CNN is even larger in Figure 5.13. Comparing the MSER applied in fast R-CNN and MSER applied in R-CNN, miss-rate/FPPI curves are far apart, denoting that MSER applied in fast R-CNN outperforms MSER applied in R-CNN. The comparison among different networks also leads to a consistent conclusion with the result of PR curve.

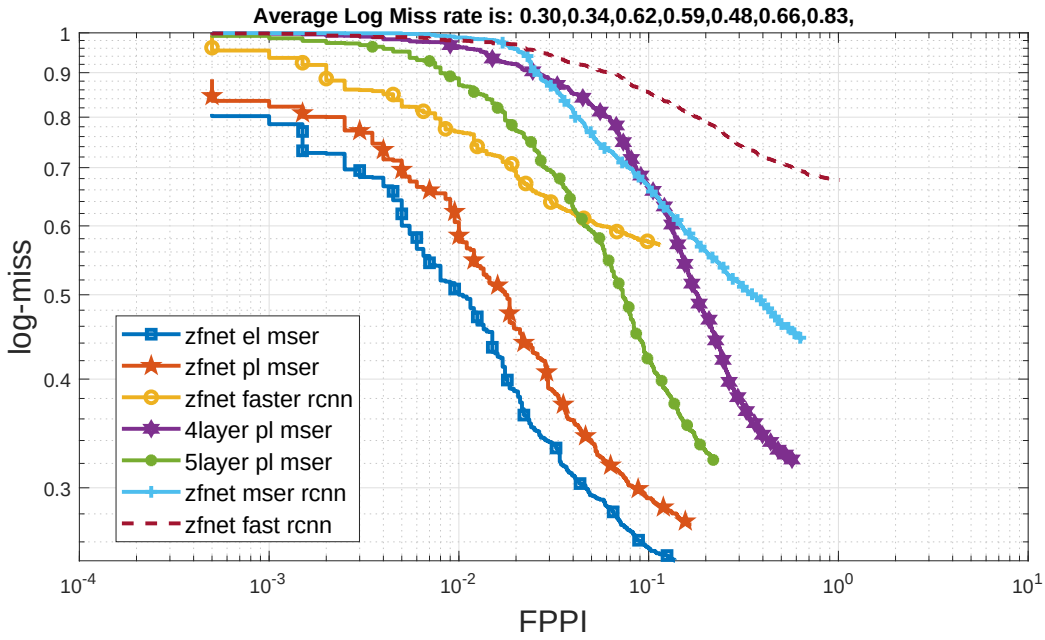


Figure 5.12: Miss-Rate/FPPI curves among different algorithms for all categories.

Table 5.3: Logarithm average miss rate.

algorithm	moose	deer	elk	horse	all
ZFnet El MSER fast R-CNN	**0.20**	**0.27**	**0.34**	**0.54**	**0.30**
ZFnet Pl MSER fast R-CNN	0.25	0.29	0.39	0.55	0.34
ZFnet faster R-CNN	0.57	0.45	0.53	0.71	0.62
4-layer PL MSER R-CNN	0.51	0.54	0.63	0.78	0.59
5-layer PL MSER R-CNN	0.37	0.41	0.47	0.68	0.48
ZFNet MSER R-CNN	0.48	0.72	0.73	0.86	0.66
ZFNet fast R-CNN	0.81	0.83	0.83	0.79	0.83

To be added, we notice that the detection achieves the best improvement for the moose category as shown in both average precision and mean average logarithm miss rate. The detection in horse, however, performs the worst compared to other categories. The reason for such phenomenon may result from the region proposal. As shown in 5.17, the curve of recall/IoU-Threshold is much higher than other categories. It is more likely for our region proposal method to extract moose regions out. The possible cause is that the pixels of moose take more apparent and different regions in intensity from their background.

5.3.3 Analysis of IoU

As mentioned in former sections, IoU larger than 50% is regarded as a qualified detection. When IoU is larger than 0% but smaller than 50%, the detection will be rejected. Based on the observation and analysis to the result falling in this range, we separate the images

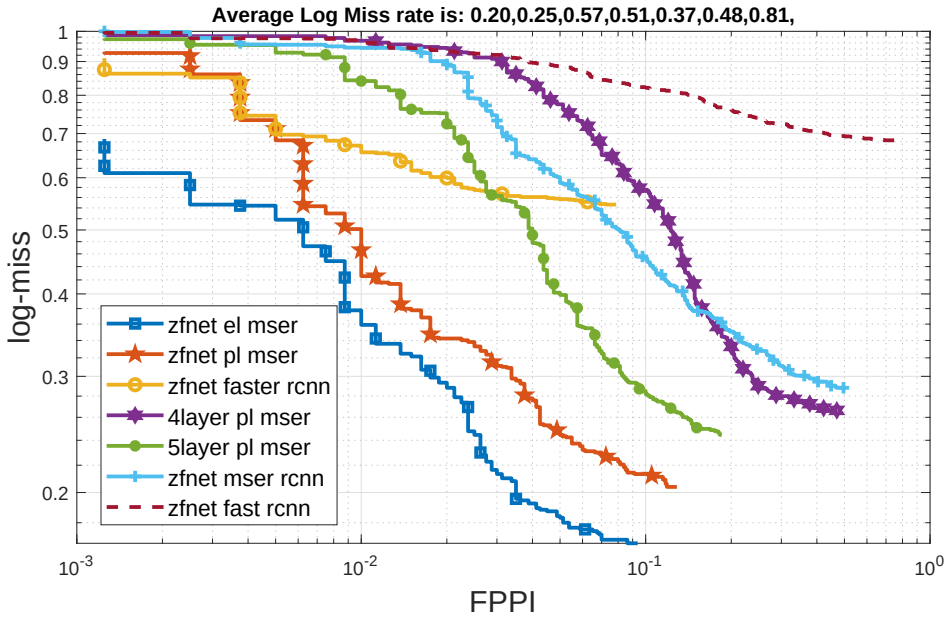


Figure 5.13: Miss rate comparison of Moose

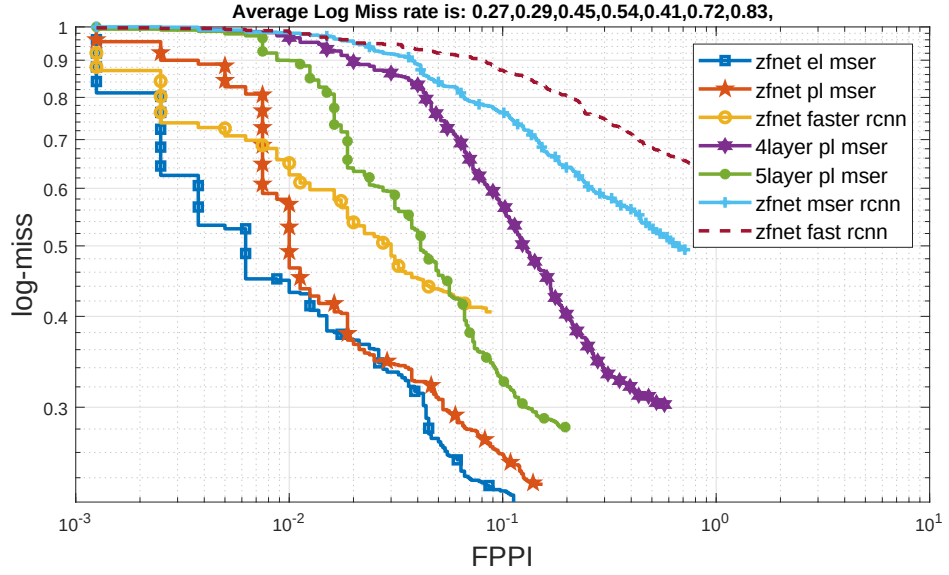


Figure 5.14: Miss rate comparison of Deer

into two cases. The first case is that the tested box is smaller than the expected one. This occurs when only part of the animal is detected that the overlap ratio is relatively low. As shown in Figure 5.18a, only the head is detected. Other examples include detecting only the horn, ear, body part or legs. The second case is that the tested box is larger than the expected one. When a larger bounding box including the expected bounding box is obtained, the IoU value is small, as shown in the example of Figure 5.18b.

The worst situation in localization occurs when the IoU is 0% denoting that animals

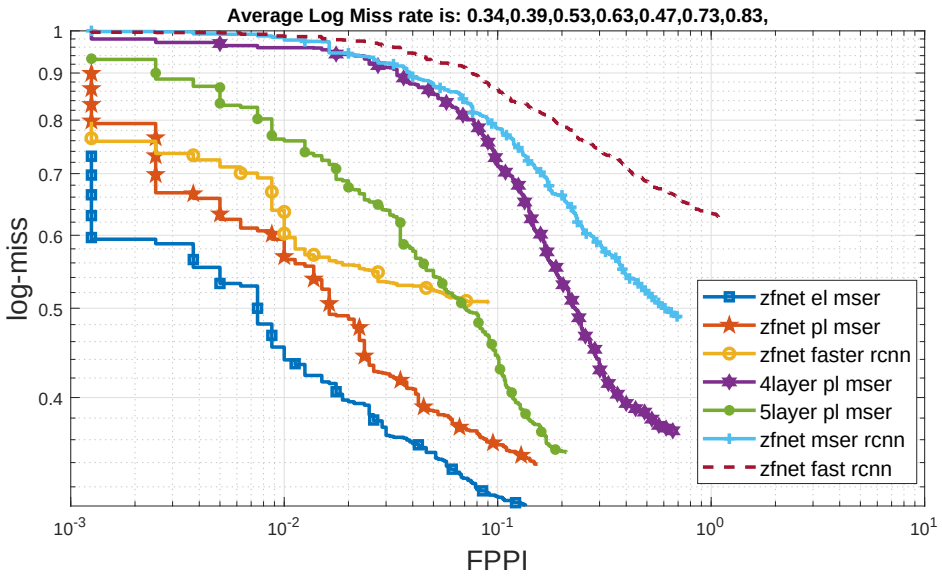


Figure 5.15: Miss rate comparison of Elk

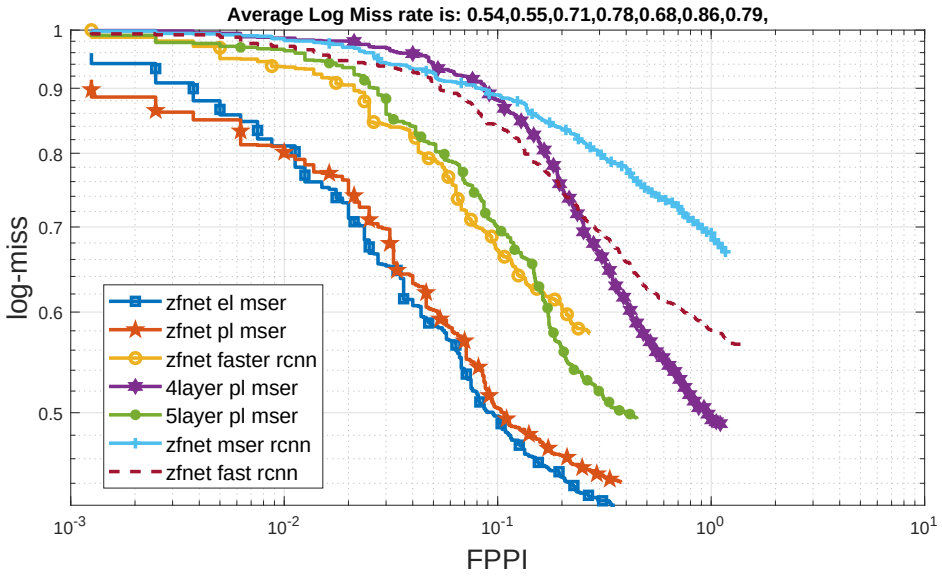


Figure 5.16: Miss rate comparison of Horse

are not detected. There could be two cases as well. The first case occurs when the animal is not found. This always results from a bad MSER detection, as the method depends heavily on the resulted regions of MSER. If the animal cannot be segmented from the background, the result would be bad as the instance in figure 5.19a. The other case is the animal is located in a wrong position when a wrong object is detected as animal, say a car or a tree as shown in figure 5.19b.

To evaluate the quality of the detection and the distribution of these two unaccepted detection, we contribute to plot the histogram of the detection results by their IoU. By

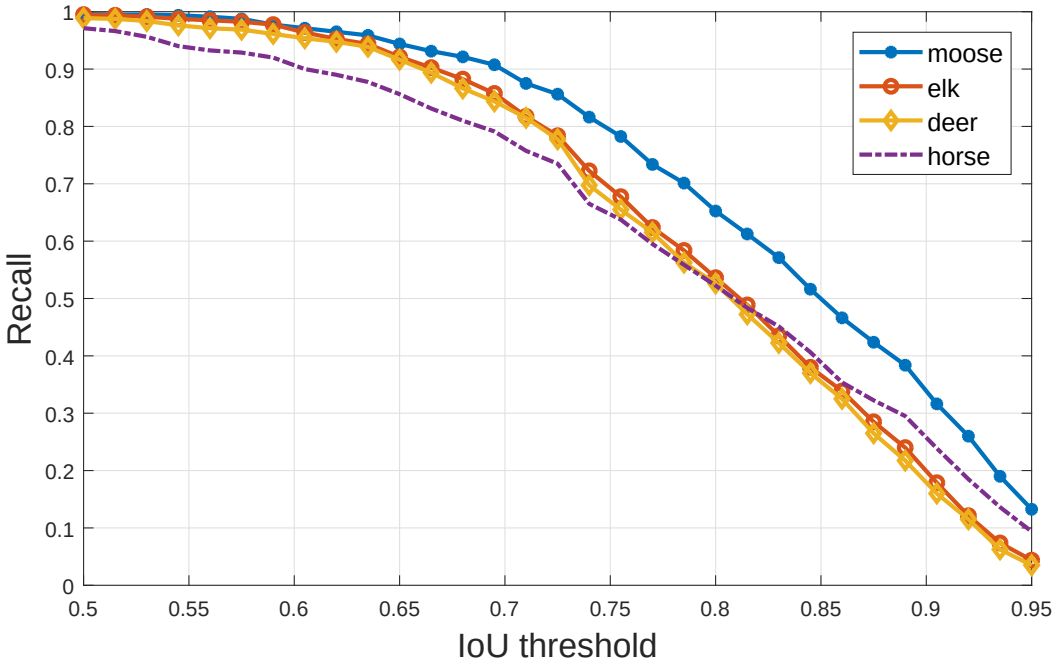


Figure 5.17: Proposed regions of different categories.

(a) (b)

Figure 5.18: Unacceptable detection. (a) Only part of the animal is detected. (b) The bounding box is too large with respect to the ground truth bounding box.

counting the IoU results of the tested images randomly selected from the dataset, we can plot the histogram of IoU. The counting is executed with respect to the actual result bounding box. That is to say, for each detection result, we calculate its IoU and finally obtain all the IoUs.

Figure 5.20 shows the IoU histogram of the detection result of all categories. In the histograms, the lines represent the proposed algorithms of Pl MSER and El MSER performs better than others in two ways. Firstly, in all histograms, the two MSER fast R-CNN lines have a lower value in the range of $[0, 0.15]$ while faster R-CNN and fast R-CNN both take

(a) (b)

Figure 5.19: IoU is 0. (a) Animal is not found. (b) Wrong position located. The tested box is locating the back bottom part of the car.

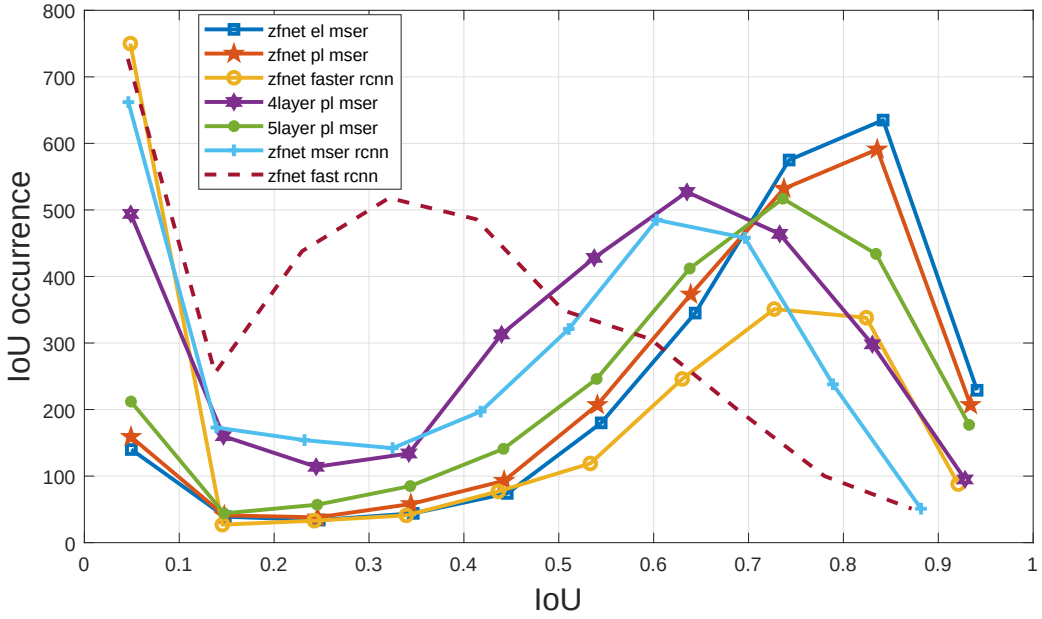


Figure 5.20: IoU histogram among different algorithms for all categories.

higher starting point. This denotes that there are less detection results evaluated as bad IoU, i.e. there are less missing detection. Secondly, the curve of the proposed algorithm has a higher or at least similar hump compared to faster R-CNN and fast R-CNN. For example, the humps in the range $[0.7, 0.9]$ of the PL MSER and EL MSER lines in Figure 5.20 are much higher than faster R-CNN and fast R-CNN, denoting more qualified detection from MSER fast R-CNN.

MSER region proposal in fast R-CNN performs better than in MSER in R-CNN can be told from the IoU histogram as well. The humps from the MSER fast R-CNN are much more to the right, implying that more acceptable IoU are obtained from the proposed

MSER regions. Humps of PL and EL MSER in Figure 5.20 are in the range of [0.7, 0.9] while the hump of MSER R-CNN is in the range of [0.5, 0.8]. All the analysis of IoU are consistent with that of PR and miss rate.

5.3.4 Analysis of Efficiency

In terms of efficiency, the processing time is recorded for comparison. For analysis, the time cost is inverted to Frame Per Second (FPS) to denote the processing speed of the algorithm. The size of the frame is of largest size 512×512. As shown in Figure 5.21, the FPS is sorted in $y - axis$ for all used detection methods.

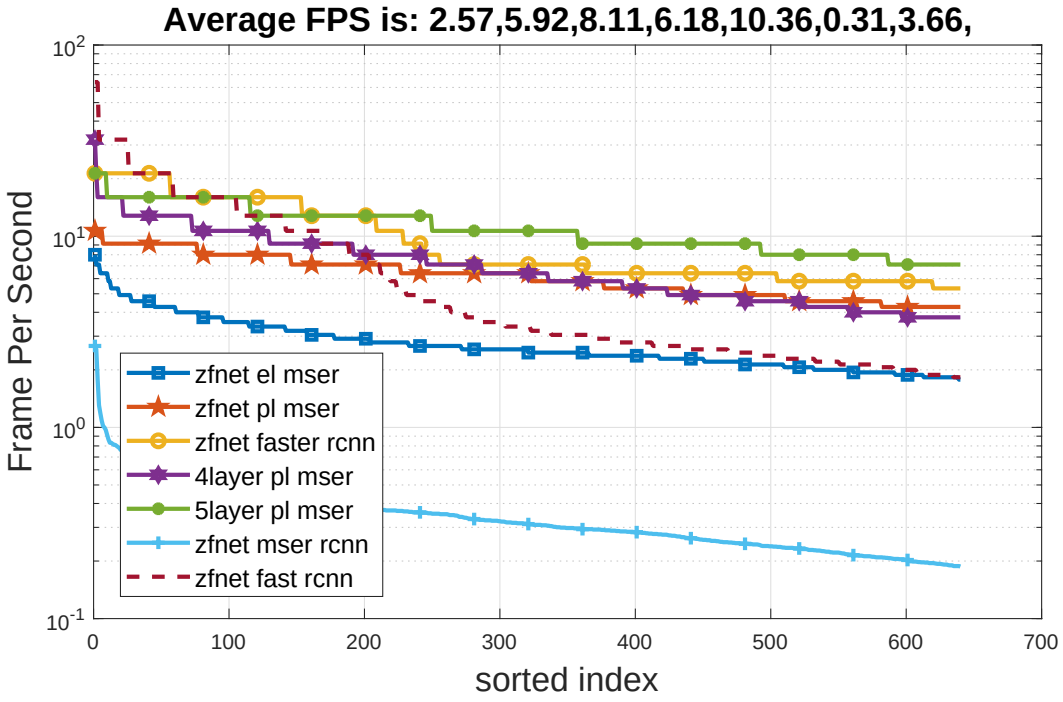


Figure 5.21: Efficiency comparison for different detection methods.

Compared to faster R-CNN whose average FPS is 8.11, the detection using EL MSER and PL MSER and ZFNet achieves smaller FPS at 2.57 and 5.92 respectively including all steps on a GPU. That is to say, the proposed detection algorithm consumes more time in each frame than faster R-CNN. The comparison result is understandable as faster R-CNN shares convolutional features between RPN and the following fast R-CNN network. Specifically, PL MSER applied in ZFNet fast R-CNN is faster than EL MSER in efficiency. The reason is that more pre-processing steps are executed in EL MSER.

In addition, when applying MSER region proposed in fast R-CNN using network with less layers than ZFNet, time consumption is saved. The FPS for 4-layer and 5-layer method using PL MSER are 6.18 and 10.36 respectively. The 5-layer method even achieves higher FPS than faster R-CNN.

Animal detection on videos can be performed to frames of video using the proposed algorithms by detecting every other multiple frames. The proposed detection method still cannot be real-time for all frames in video. Combining tracking techniques, the detection for real-time video can be achieved.

Chapter 6

Conclusion

This chapter concludes the thesis with a summary of contributions and future work.

6.1 Contributions

This thesis is a research on animal detection for intelligent vehicles. By looking back the developing history for object detection and image classification, especially the techniques based on Convolutional Neural Network, we propose to apply MSER as a novel region proposal method in the framework of R-CNN and Fast R-CNN. We create a new dataset of large animals to evaluate the detection method.

As a blob/region detection method in linear time, MSER consists two forms of output. Based on the output in form of list of pixels, PL MSER encloses regions by minimum rectangles. Using the output of ellipses which have the same first and second moments with the corresponding regions, EL MSER fits the elliptical regions to bounding boxes. Some pre-process steps are then performed to the proposed bounding boxes, including scoring by aspect-ratio, reducing overlapping regions and enlarging the modified bounding box proposals.

Applying the generated bounding boxes in R-CNN, we obtain a new framework for animal detection. R-CNN performs slow as it needs to compute CNN features for every region without sharing any computation. We achieve a better architecture by employing MSER in fast R-CNN framework. Through the intermediate ROI pooling layer, feature maps among all proposed regions are shared.

We evaluate the proposed regions by PL MSER and EL MSER using recall/IoU-Threshold curve. The proposed methods generate less regions and keep higher recall comparing to Edge-Box. PL MSER and EL MSER also performs better in recall than RPN. Through multiple experiments in our dataset of large animals, the proposed detection methods in PL MSER and EL MSER applied in fast R-CNN outperforms faster R-CNN in both precision and recall. In terms of the quality of detection, the bounding boxes resulted from PL MSER and EL MSER achieve higher Intersection over Union (IoU).

Concerning the efficiency, we succeed in speeding up the detection by reducing the number of regions 2 to 5 times less than the region proposal method of Edge-Box.

6.2 Future work

In terms of detection speed, the proposed framework is slower than faster R-CNN using same network from the experiments result. The future work should focus on cutting down the computation cost in MSER procedure.

There is another deficiency to our detection due to the drawback of MSER region proposal method. MSER is good at detecting a complete region for object which has small changes in gray-level intensities or intensities from different color channels. Nevertheless, for animals which have multiple main color such as panda or zebra, MSER detection method does not perform well.

The real-time application of MSER on smart vehicles has some room for improvement. On one hand, the hardware design for MSER algorithm is rare and limited. On the other hand, the region proposal method is yet to be validated on object detection task of other categories, such as human and vehicles.

References

[1] Kaouther Abrougui, Azzedine Boukerche, and Richard Werner Nelem Pazzi. Design and evaluation of context-aware and location-based service discovery protocols for vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, 12(3):717–735, 2011.

[2] Osama Abumansoor and Azzedine Boukerche. A secure cooperative approach for nonline-of-sight location verification in vanet. *IEEE Transactions on Vehicular Technology*, 61(1):275–285, 2012.

[3] Elie El Ajaltouni, Azzedine Boukerche, and Ming Zhang. An efficient dynamic load balancing scheme for distributed simulations on a grid infrastructure. In *Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, pages 61–68. IEEE Computer Society, 2008.

[4] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202, Nov. 2012.

[5] Thanasis Antoniou, Ioannis Chatzigiannakis, George Mylonas, Sotiris Nikoletseas, and Azzedine Boukerche. A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range. In *Proceedings of the 37th annual symposium on Simulation*, page 43. IEEE Computer Society, 2004.

[6] Oren Barkan, Jonathan Weill, Lior Wolf, and Hagai Aronowitz. Fast high dimensional vector multiplication face recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1960–1967, Dec. 2013.

[7] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision (ECCV)*, pages 404–417. Springer, May 2006.

[8] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, Apr. 2002.

[9] Gunilla Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3):344–371, June 1986.

[10] Azzedine Boukerche, Ioannis Chatzigiannakis, and Sotiris Nikoletseas. A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range. *Computer communications*, 29(4):477–489, 2006.

[11] Azzedine Boukerche and Amir Darehshoorzadeh. Opportunistic routing in wireless networks: Models, algorithms, and classifications. *ACM Computing Surveys (CSUR)*, 47(2):22, 2015.

[12] Azzedine Boukerche, Sajal K Das, Alessandro Fabbri, and Oktay Yildiz. Exploiting model independence for parallel pcs network simulation. In *Parallel and Distributed Simulation, 1999. Proceedings. Thirteenth Workshop on*, pages 166–173. IEEE, 1999.

[13] Azzedine Boukerche and Caron Dzermajko. Performance evaluation of data distribution management strategies. *Concurrency and Computation: Practice and Experience*, 16(15):1545–1573, 2004.

[14] Azzedine Boukerche and Xin Fei. A coverage-preserving scheme for wireless sensor network with irregular sensing range. *Ad hoc networks*, 5(8):1303–1316, 2007.

[15] Azzedine Boukerche, Xin Fei, and Regina B Araujo. An optimal coverage-preserving scheme for wireless sensor networks based on local information exchange. *Computer Communications*, 30(14-15):2708–2720, 2007.

[16] Azzedine Boukerche, Sungbum Hong, and Tom Jacob. An efficient synchronization scheme of multimedia streams in wireless and mobile systems. *IEEE transactions on Parallel and Distributed Systems*, 13(9):911–923, 2002.

[17] Azzedine Boukerche, Anahit Martirosyan, and Richard Pazzi. An inter-cluster communication based energy aware and fault tolerant protocol for wireless sensor networks. *Mobile Networks and Applications*, 13(6):614–626, 2008.

[18] Azzedine Boukerche, Nathan J McGraw, Caron Dzermajko, and Kaiyuan Lu. Grid-filtered region-based data distribution management in large-scale distributed simulation systems. In *Simulation Symposium, 2005. Proceedings. 38th Annual*, pages 259–266. IEEE, 2005.

[19] Azzedine Boukerche, Horacio ABF Oliveira, Eduardo F Nakamura, and Antonio AF Loureiro. Localization systems for wireless sensor networks. *IEEE wireless Communications*, 14(6), 2007.

[20] Azzedine Boukerche, Richard Werner N Pazzi, and Regina B Araujo. Hpeq a hierarchical periodic, event-driven and query-based wireless sensor network protocol. In *null*, pages 560–567. IEEE, 2005.

[21] Azzedine Boukerche, Richard WN Pazzi, and Jing Feng. An end-to-end virtual environment streaming technique for thin mobile devices over heterogeneous networks. *Computer Communications*, 31(11):2716–2725, 2008.

[22] Azzedine Boukerche and Yonglin Ren. A security management scheme using a novel computational reputation model for wireless and mobile ad hoc networks. In *Proceedings of the 5th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 88–95. ACM, 2008.

[23] Azzedine Boukerche and Yonglin Ren. A trust-based security system for ubiquitous and pervasive computing environments. *Computer Communications*, 31(18):4343–4351, 2008.

[24] Azzedine Boukerche, Cristiano Rezende, and Richard W Pazzi. Improving neighbor localization in vehicular ad hoc networks to avoid overhead from periodic messages. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pages 1–6. IEEE, 2009.

[25] Azzedine Boukerche and Steve Rogers. Gps query optimization in mobile and wireless networks. In *Computers and Communications, 2001. Proceedings. Sixth IEEE Symposium on*, pages 198–203. IEEE, 2001.

[26] Azzedine Boukerche and Amber Roy. Dynamic grid-based approach to data distribution management. *Journal of Parallel and Distributed Computing*, 62(3):366–392, 2002.

[27] Azzedine Boukerche, Amber Roy, and Neville Thomas. Dynamic grid-based multicast group assignment in data distribution management. In *ds-rt*, page 47. IEEE, 2000.

[28] Azzedine Boukerche and Samer Samarah. A novel algorithm for mining association rules in wireless ad hoc sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 19(7):865–877, 2008.

[29] Azzedine Boukerche and Carl Tropper. A distributed graph algorithm for the detection of local cycles and knots. *IEEE Transactions on Parallel and Distributed Systems*, 9(8):748–757, 1998.

[30] Azzedine Boukerche and Damla Turgut. Secure time synchronization protocols for wireless sensor networks. *IEEE Wireless Communications*, 14(5), 2007.

[31] Tilo Burghardt and Janko Calic. Real-time face detection and tracking of animals. In *8th Seminar on Neural Network Applications in Electrical Engineering*, pages 27–32, Sept. 2006.

[32] Transport Canada. Update of data sources on collisions involving motor vehicles and large animals in canada. [Online]. Available: https://www.tc.gc.ca/eng/motorvehiclesafety/tp-tp14798-menu-160.htm. Accessed on: Jan. 2018.

[33] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, Nov. 1986.

[34] Mingming Cheng, Ziming Zhang, Wenyan Lin, and Philip Torr. Bing: Binarized normed gradients for objectness estimation at 300fps. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3286–3293, June 2014.

[35] Dmitry Chetverikov and Yuri Khenokh. Matching for shape defect detection. In *8th International Conference on Computer Analysis of Images and Patterns (CAIP)*, pages 367–374, Sept. 1999.

[36] JS Christie and S Nason. Analysis of vehicle collisions with moose and deer on new brunswick arterial highways. In *The 31st Annual Conference of the Canadian Society for Civil Engineering, Moncton, New Brunswick. 11p*, pages 652–661, 2003.

[37] Rodolfo WL Coutinho, Azzedine Boukerche, Luiz FM Vieira, and Antonio AF Loureiro. Geographic and opportunistic routing for underwater sensor networks. *IEEE Transactions on Computers*, 65(2):548–561, 2016.

[38] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. Cambridge University Press, New York, NY, USA, 2000.

[39] Felipe Cunha, Leandro Villas, Azzedine Boukerche, Guilherme Maia, Aline Viana, Raquel AF Mini, and Antonio AF Loureiro. Data communication in vanets: Protocols, applications and challenges. *Ad Hoc Networks*, 44:90–103, 2016.

[40] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893, June 2005.

[41] Horacio Antonio Braga Fernandes De Oliveira, Azzedine Boukerche, Eduardo Freire Nakamura, and Antonio Alfredo Ferreira Loureiro. An efficient directed localization recursion protocol for wireless sensor networks. *IEEE Transactions on Computers*, (5):677–691, 2008.

[42] Rachid Deriche. Using canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, 1(2):167–187, June 1987.

[43] Piotr Dollár. Piotr's Computer Vision Matlab Toolbox (PMT). [Online]. Available: https://github.com/pdollar/toolbox. Accessed on: Jan. 2018.

[44] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761, Apr. 2012.

[45] Piotr Dollár and C. Lawrence Zitnick. Structured forests for fast edge detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1841–1848, Dec. 2013.

[46] Piotr Dollár and C Lawrence Zitnick. Fast edge detection using structured forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8):1558–1570, Aug. 2015.

[47] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, Oct. 2012.

[48] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. Performance analysis of a distributed comparison-based self-diagnosis protocol for wireless ad-hoc networks. In *Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*, pages 165–172. ACM, 2006.

[49] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. A distributed fault identification protocol for wireless and mobile ad hoc networks. *Journal of Parallel and Distributed Computing*, 68(3):321–335, 2008.

[50] Li Fei-Fei and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 524–531, June 2005.

[51] 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, Sept. 2010.

[52] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, Sept. 2004.

[53] Per-Erik Forssén. Maximally stable colour regions for recognition and matching. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2007.

[54] Per-Erik Forssén and David G Lowe. Shape descriptors for maximally stable extremal regions. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 59–73, Oct. 2007.

[55] Dariu M Gavrila. A bayesian, exemplar-based approach to hierarchical shape matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1408–1421, Aug. 2007.

[56] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, Dec. 2015.

[57] 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 (CVPR)*, pages 580–587, June 2014.

[58] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158, Jan. 2016.

[59] Ross Girshick, Forrest Iandola, Trevor Darrell, and Jitendra Malik. Deformable part models are convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 437–446, June 2015.

[60] Ross Brook Girshick. *From Rigid Templates to Grammars: Object Detection with Structured Models*. PhD thesis, University of Chicago, Chicago, IL, USA, 2012.

[61] Chunhui Gu, Joseph J Lim, Pablo Arbeláez, and Jitendra Malik. Recognition using regions. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1030–1037, June 2009.

[62] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50, Sept. 1988.

[63] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision (ECCV)*, pages 346–361. Springer, Sept. 2014.

[64] 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 (CVPR)*, pages 770–778, June 2016.

[65] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. In *European Conference on Computer Vision (ECCV)*, pages 340–353. Springer, Oct. 2012.

[66] Jan Hosang, Rodrigo Benenson, Piotr Dollár, and Bernt Schiele. What makes for effective detection proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):814–830, Apr. 2016.

[67] Jan Hendrik Hosang, Rodrigo Benenson, and Bernt Schiele. How good are detection proposals, really? *Computing Research Repository (CoRR)*, abs/1406.6962, 2014.

[68] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187, Feb. 1962.

[69] Marcel P Huijser, Patrick Tracy McGowen, Julie Fuller, Amanda Hardy, and A Kociolek. Wildlife-vehicle collision reduction study: Report to congress. Technical report, 2007.

[70] Daniel P. Huttenlocher, Gregory A. Klanderman, and William J Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, Sep. 1993.

[71] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, Jul. 2015.

[72] Luo Juan and Oubong Gwun. A comparison of sift, pca-sift and surf. *International Journal of Image Processing (IJIP)*, 3(4):143–152, 2009.

[73] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, Dec. 2012.

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

[75] Feifei Li. Cs231n convolutional neural networks for visual recognition. Accessed on: Jan. 2018.

[76] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *Computing Research Repository (CoRR)*, abs/1312.4400, 2013.

[77] Haibin Ling and David W Jacobs. Using the inner-distance for classification of articulated shapes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 719–726, June 2005.

[78] Haibin Ling and David W Jacobs. Shape classification using the inner-distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):286–299, Feb. 2007.

[79] 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 (ECCV)*, pages 21–37. Springer, Oct. 2016.

[80] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157, Sept. 1999.

[81] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004.

[82] Abdelhamid Mammeri, Depu Zhou, and Azzedine Boukerche. Animal-vehicle collision mitigation system for automated vehicles. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(9):1287–1299, Sept. 2016.

[83] David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, May 2004.

[84] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, Sept. 2004.

[85] Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 525–531, July 2001.

[86] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, Oct. 2004.

[87] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72, Nov. 2005.

[88] PE Mohammad Ashkan Sharafsaleh, Marcel Huijser, Christopher Nowakowski, Mark C Greenwood, Larry Hayden, Jonathan Felder, and May Wang. Evaluation of an animal warning system effectiveness phase two-final report. 2012.

[89] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60, Nov. 1996.

[90] WL Myers, WY Chang, SS Germaine, WM Vander Haegen, and TE Owens. An analysis of deer and elk-vehicle collision sites along state highways in washington state. *Completion Report. Olympia, WA, USA: Washington Department of Fish and Wildlife*, 701, 2008.

[91] David Nistér and Henrik Stewénius. Linear time maximally stable extremal regions. In *European Conference on Computer Vision (ECCV)*, pages 183–196. Springer, Oct. 2008.

[92] David Obdržálek, Stanislav Basovník, Lukáš Mach, and Andrej Mikulík. Detecting scene elements using maximally stable colour regions. In *International Conference on Research and Education in Robotics (EUROBOT)*, pages 107–115, May 2009.

[93] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, Jul. 2002.

[94] Horacio ABF Oliveira, Eduardo F Nakamura, Antonio AF Loureiro, and Azzedine Boukerche. Error analysis of localization systems for sensor networks. In *Proceedings of the 13th annual ACM international workshop on Geographic information systems*, pages 71–78. ACM, 2005.

[95] Richard W Pazzi and Azzedine Boukerche. Propane: A progressive panorama streaming protocol to support interactive 3d virtual environment exploration on graphics-constrained devices. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 11(1):5, 2014.

[96] Michal Perdoch, Jiri Matas, and Stepan Obdrzalek. Stable affine frames on isophotes. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, Oct. 2007.

[97] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3):61–74, 1999.

[98] The Wildlife Collision Prevention Program. The facts of wildlife vehicle collisions: A hard hit. [Online]. Available: http://www.wildlifecollisions.ca/thefacts.htm. Accessed on: Jan. 2018.

[99] Deva Ramanan, David A Forsyth, and Kobus Barnard. Building models of animals from video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1319–1334, Aug. 2006.

[100] 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 (CVPR)*, pages 779–788, June 2016.

[101] 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 (NIPS)*, pages 91–99, Dec. 2015.

[102] Cristiano G Rezende, Azzedine Boukerche, Heitor S Ramos, and Antonio AF Loureiro. A reactive and scalable unicast solution for video streaming over vanets. *IEEE Trans. Computers*, 64(3):614–626, 2015.

[103] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. *Computer Vision–ECCV 2006*, pages 430–443, 2006.

[104] Edward Rosten, Reid Porter, and Tom Drummond. Machine learning for high-speed corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–119, Jan. 2010.

[105] William J Rucklidge. Efficiently locating objects using the hausdorff distance. *International Journal of Computer Vision*, 24(3):251–270, Sept. 1997.

[106] Samer Samarah, Muhannad Al-Hajri, and Azzedine Boukerche. A predictive energy-efficient technique to support object-tracking sensor networks. *IEEE Transactions on Vehicular Technology*, 60(2):656–663, 2011.

[107] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–535, May 1997.

[108] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, June 2000.

[109] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[110] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Computing Research Repository (CoRR)*, abs/1409.1556, 2014.

[111] 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 (CVPR)*, pages 1–9, June 2015.

[112] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2553–2561, Dec. 2013.

[113] Yichuan Tang. Deep learning using linear support vector machines. *Computing Research Repository (CoRR)*, abs/1306.0239, 2013.

[114] Zongzhi Tang. *A Novel Road Marking Detection and Recognition Technique Using a Camera-based Advanced Driver Assistance System.* PhD thesis, Université d'Ottawa/University of Ottawa, 2017.

[115] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: A survey. *Found. Trends. Comput. Graph. Vis.*, 3(3):177–280, Jul. 2008.

[116] 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, Sept. 2013.

[117] Andrea Vedaldi and Brian Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pages 1469–1472, New York, NY, USA, 2010. ACM.

[118] Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, pages 689–692, New York, NY, USA, 2015. ACM.

[119] Leandro Villas, Azzedine Boukerche, Regina Borges De Araujo, and Antonio AF Loureiro. Highly dynamic routing protocol for data aggregation in sensor networks. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*, pages 496–502. IEEE, 2010.

[120] Leandro A Villas, Azzedine Boukerche, Daniel L Guidoni, Horacio ABF De Oliveira, Regina Borges De Araujo, and Antonio AF Loureiro. An energy-aware spatio-temporal correlation mechanism to perform efficient data collection in wireless sensor networks. *Computer Communications*, 36(9):1054–1066, 2013.

[121] Leandro A Villas, Daniel L Guidoni, Regina B Araújo, Azzedine Boukerche, and Antonio AF Loureiro. A scalable and dynamic data aggregation aware routing protocol for wireless sensor networks. In *Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems*, pages 110–117. ACM, 2010.

[122] Leandro Aparecido Villas, Azzedine Boukerche, Guilherme Maia, Richard Werner Pazzi, and Antonio AF Loureiro. Drive: An efficient and robust data dissemination protocol for highway and urban vehicular ad hoc networks. *Computer Networks*, 75:381–394, 2014.

[123] Leandro Aparecido Villas, Azzedine Boukerche, Heitor Soares Ramos, Horacio AB Fernandes de Oliveira, Regina Borges de Araujo, and Antonio Alfredo Ferreira Loureiro. Drina: A lightweight and reliable routing approach for in-network aggregation in wireless sensor networks. *IEEE Transactions on Computers*, 62(4):676–689, 2013.

[124] Bing Wang and ShaoSheng Fan. An improved canny edge detection algorithm. In *Second International Workshop on Computer Science and Engineering (WCSE)*, pages 497–500, Oct. 2009.

[125] Mingqiang Yang, Kidiyo Kpalma, and Joseph Ronsin. A Survey of Shape Feature Extraction Techniques. In *Pattern Recognition*, pages 43–90. IN-TECH, 2008.

[126] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833. Springer, Sept. 2014.

[127] Dengsheng Zhang and Guojun Lu. Review of shape representation and description techniques. *Pattern Recognition*, 37(1):1–19, Jan. 2004.

[128] Zhenxia Zhang, Richard W Pazzi, and Azzedine Boukerche. A mobility management scheme for wireless mesh networks based on a hybrid routing protocol. *Computer Networks*, 54(4):558–572, 2010.

[129] Debao Zhou, Jingzhou Wang, and Shufang Wang. Countour based hog deer detection in thermal images for traffic safety. In *Proceedings of the 2012 International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*, pages 969–974, Jul. 2012.

[130] Depu Zhou. *Real-time animal detection system for intelligent vehicles*. PhD thesis, Université d'Ottawa/University of Ottawa, 2014.

[131] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision (ECCV)*, pages 391–405. Springer, Sept. 2014.

Appendix A

MSER algorithm

With the basic definitions defined in 3.3.1, some structures are used in the modified watershed algorithm [91].

1. For pixels which have been already accessed by the water, a binary mask is used to represent the accessible pixels or flooded pixels.
2. There are pixels which the water has access to but not yet enter and explore all the edges out. Denoting the priority of flooding by the minus of the intensity, a priority queue B is defined for theses pixels partially flooded. Along with the pixel id, an edge number indicating the next edge to be explored can be stored in the queue.
3. The component stack C is used to store the component information, where each element holds the pixels in the component and the size history of the component, as well as the intensity at which the component is being processed. The intensity level is used to represent the stack.

The detailed MSER implementation is shown in algorithm 1. The input of the algorithm is a gray image and the output is the component stack with each component a list of pixels. The algorithms is initialized with an empty component stack and an empty heap of boundary pixels. The component stack has only one dummy component with intensity level higher than any allowed in the image. The heap stores the boundary pixels of a flooded region. The intensity level of boundary pixels should not be lower than the region pixels.

In this algorithm, the upper left corner pixel is usually picked as source pixel. After the loop of pushing neighbors onto the stack, the number of pixels in component is actually the area of the region.

Algorithm 1 MSER Algorithm

Initialization: clean stack C of components, heap B of boundary pixels and accessible mask.
Pick any source pixel as *current pixel*. Mark the *current pixel* as accessible and set the intensity as $current_{level}$.
Push an empty component with $current_{level}$ onto stack C.
do
  while *current pixel* has accessible neighbors in 4-neighborhood **do**
    $neighbor_{level} \leftarrow$ the intensity of the neighbor pixel
    if $neighbor_{level} \geq current_{level}$ **then**
      Push the **neighbor pixel** onto the heap of boundary pixels B.
    else
      Push the **current pixel** back onto the heap of boundary pixels B;
      current pixel $\leftarrow$ **neighbor pixel**;
      Push an empty component with $current_{level}$ onto stack C.
    end if
  end while
  Count the current number of pixels in the component at the top of the stack C.
  if The heap of boundary pixels is empty **then**
    return;
  else
    Pop the heap B and get the intensity new_{level}.
    if $new_{level} > current_{level}$ **then**
      do
        Obtain the intensity of the second component on the stack: snd_{level}
        if $new_{level} < snd_{level}$ **then**
          $TopStack_{level} = new_{level}$; continue;
        else
          Merge top-two components on stack C
        end if
      while $new_{level} > TopStack_{level}$
    end if
  end if
while The heap of boundary pixels is empty or the dummy component is popped

Appendix B

Edge box algorithm

The implementation of edge box can be expressed in the following steps.

1. **Search for candidate bounding boxes.** To find the candidate bounding boxes, edge box uses sliding window search over position, scale and ratio. The step size parameter α is used to denote the IoU with neighboring boxes such that one step results in neighboring box with IoU α in translation, scale and aspect ratio. $\alpha = 0.65$ in practice. The scale is controlled by the area range from $\sigma = 1000$ to the full image. The aspect ratio varies from $1/\tau$ to τ and $\tau = 3$.

2. **Obtain edge-response map.** The edge map is computed from original image using Structured Edge detector [43, 45]. A non-maximal suppression is then performed to obtain a sparse edge map from the dense edge responses. In the resulted map, each pixel p has an edge magnitude m_p and orientation θ_p, where pixels with $m_p > 0.1$ are defined as edges.

3. **Compute *edge groups*.** An edge group is formed by gathering the edge points which are approximately on a line. The specific method is to searching for 8-connected edge points to add in the group until the sum of their directional orientations is larger than $\pi/2$. Small groups are merged with neighboring groups. The result of this step is edge group set denoted by $S = s_1, ..., s_n$.

4. **Calculate the affinity between each pair of neighboring groups.** For each pair $s_i \in S$ and $s_j \in S$, the affinity $a(i, j)$ is computed by B.1 based on their mean positions x_i and x_j and mean orientations θ_i and θ_j.

$$a(s_i, s_j) = |\cos(\theta_i - \theta_{ij}) \cos(\theta_j - \theta_{ij})|^\gamma \tag{B.1}$$

   Where θ_{ij} denotes the angel between x_i and x_j. The value of γ adjusts the affinity's sensitivity to changes in orientation. The value of γ is set to 2 in implementation.

   If two edge groups are separated by more than 2 pixels their affinity is set to zero. For increased computational efficiency only affinities above a small threshold (0.05) are stored and the rest are assumed to be zero. Edge groups that are on one line tends to achieve higher similarity by the formula of B.1.

5. **Find the intersection of edge groups.** To find intersections along a horizontal boundary in row r, two data structures L_r and K_r are created for each row of the image to find the list of overlapping edge groups efficiently. L_r is an ordered list which stores the index of the edge group in row r. An index is only added to L_r if the edge group index changes from one pixel to the next. For pixels that are not edges between edge groups, a zero is added to L_r. As multiple pixels in the row may be belong to the same edge group or not edges, the size of L_r is much smaller than the width of the image. The other data structure K_r, however, has the same size as the width of the image. K_r is created by the corresponding index in L_r for each column c in row r. For instance, pixel p at location (c, r) is a member of edge group s_i, $L_r(K_r(c)) = i$. For the horizontal boundary from pixel (c_0, r) to (c_n, r), the list of overlapping edge groups can be efficiently found by searching L_r from index $K_r(c_0)$ to $K_r(c_1)$.

6. **Score the candidate bounding boxes.** For each group $s_i \in S$, a weight $w_b(s_i)$ is calculated to indicate whether s_i is wholly enclosed in bounding box b. s_i is wholly contained in b when $w_b(s_i) = 1$ and not when $w_b(s_i) = 0$. Two specific cases sets the weight $w_b(s_i)$ as zero. The first case is when the edge overlaps the boundary of the bounding box b. The other case is when an arbitrary pixel $\hat{x}_i$ in s_i satisfies $\hat{x}_i \notin b$. For edges beyond these two cases, if there exists a path that connects the edges from an edge which overlaps the boundary to s_i, the weight is given by Equation B.2. The symbols from t_1 to $t_{|T|}$ denotes the head and tail of the path. Otherwise, if no such path exist, $w_b(s_i) = 1$.

$$w_b(s_i) = 1 - \max_T \prod_i^{|T|-1} a(t_j, t_{j+1}) \tag{B.2}$$

Through the assignment of weight, the weight of most pairwise is set to zero so that the full step can be executed rapidly. Denoting the sum m_i of the magnitudes m_p for all edges and the computed weight, the final score for the bounding box b is given by Equation B.3. Where b_w and b_h are the width and height of the bounding box. To avoid a higher score on larger windows which have more edges on average, the parameter of $\mathcal{K}$ is used and set to 1.5 in practice.

$$h_b = \frac{\sum_i \omega_b(s_i) m_i}{2(b_w + b_h)^{\mathcal{K}}} \tag{B.3}$$

In the actual implementation, an integral image is used to speed up the computation in Equation B.3, especially for summing of m_i. Another observation is that the edges around the boundary of the bounding box is more important than those in the center. A new formula is created to achieve slightly better accuracy in B.4, where the bounding box b_{in} takes a width and height of $b_w/2$ and $b_h/2$ respectively.

$$h_b^{in} = h_b - \frac{\sum_{p \in b^{in}} m_p}{2(b_w + b_h)^{\mathcal{K}}} \tag{B.4}$$

Appendix C

Selective search algorithm

In the algorithm of selective search, initial regions $R = r_1, ..., r_n$ are firstly obtained through graph-based segmentation [52].

C.1 Image segmentation algorithm

Let $G = (V, E)$ be an undirected graph, with vertices $v \in V$ where V is the set of elements to be segmented, and edges $(v_i, v_j) \in E$ corresponding to pairs of neighboring vertices. Each edge $(v_i, v_j) \in E$ has a corresponding weight $w(v_i, v_j)$, which is a non-negative measure of the dissimilarity between neighboring elements v_i and v_j.

In the case of image segmentation, the elements $v \in V$ are pixels and the weight $w(v_i, v_j)$ is some measure of the dissimilarity between the two pixels v_i and v_j connected by that edge $(v_i, v_j) \in E$. The dissimilarity could be measured by the difference in intensity, color, motion, location or some other local attribute. A segmentation S is a partition of V so that each component or region $C \in S$ corresponds to a connected component in a graph $G' = (V, E')$, $E' \subseteq E$.

C.2 Selective Search

As shown in Algorithm 3, after initial regions $R = r_1, ..., r_n$ are obtained, for region r_i and its neighboring region r_j, the similarity $s(r_i, r_j)$ is calculated by four different similarity measures. The similarity measures are complementary and fast-to-compute including s_{color}, $s_{texture}$, s_{size} and s_{fill}. The final similarity measure is given by Equation C.1.

$$s(r_i, r_j) = a_1 s_{color}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j) \tag{C.1}$$

Separately, s_{color} measures color similarity using the intersection between the one-dimensional histograms for all three color channels. $s_{texture}$ measures texture using the histogram intersection between SIFT-like texture measurements. s_{size} encourages small

Algorithm 2 Segmentation Algorithm

Input: A graph $G = (V, E)$, with n vertices and m edges
Output: A segmentation of V into components $S = (C_1, ..., C_r)$.
Sort E into $\pi = o_1, ..., o_m$ by non-decreasing edge weight.
Initialize with a segmentation S_0, where each vertex v_i is in its own component.
while $q \in 1, ..., m$ **do**
For $o_q = v_i, v_j$, we use C_i^{q-1} to denote component of S^{q-1} containing v_i and C_j^{q-1} containing v_j.
if then$C_i^{q-1} \neq C_j^{q-1}$ and $w(o_q) \leq MInt(C_i^{q-1}, C_j^{q-1})$ $S_q = S_{q-1} \cup (C_i^{q-1}, C_j^{q-1})$
else$S_q = S_{q-1}$
end if
end while
$S = S^m$

Algorithm 3 Hierarchical Grouping Algorithm

Input: (color) Image
Output: Set of object locations L
Obtain initial regions $R = r_1, ..., r_n$ through graph-based segmentation.
Initialize similarity set $S = \emptyset$
while $i, j \in [1, ..., n], i \neq j$ **do**
Calculate similarity between *Neighboring region pair* (r_i, r_j): $s(r_i, r_j)$
$S = S \cup s(r_i, r_j)$
end while
while $S \neq \emptyset$ **do**
Get highest similarity $s(r_i, r_j) = \max S$
Merge corresponding regions $r_t = r_i \cup r_j$
Remove similarities regarding $r_i : S = S\ s(r_i, r_*)$
Remove similarities regarding $r_j : S = S\ s(r_*, r_j)$
Calculate similarity set S_t between r_t and its neighbors
$S = S \cup s_t$
$R = R \cup r_t$
end while

regions to merge early by the fraction of the image that the two neighboring regions jointly occupy. s_{fill} measures how well two regions fit into each other. As the similarities can be efficiently propagated through the hierarchy, the computation can be fast.

In the following step, we get the highest similarity and merge the two corresponding regions iteratively and finally get the whole hierarchy of the proposed regions.