Interactive Object Detection

Priyanka Subramanya Vokuda

Publisher: Dean Prof. Dr. Wolfgang Heiden

Hochschule Bonn-Rhein-Sieg – University of Applied Sciences,
Department of Computer Science

Sankt Augustin, Germany

April 2019

Technical Report 01-2019

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

ISSN 1869-5272

ISBN 978-3-96043-072-8

Digital Object Identifier doi:10.18418/978-3-96043-072-8
DOI-Resolver http://dx.doi.org/

Bonn Rhein-Sieg University of Applied Sciences

Abstract

The success of state-of-the-art object detection methods depend heavily on the availability of a large amount of annotated image data. The raw image data available from various sources are abundant but non-annotated. Annotating image data is often costly, time-consuming or needs expert help. In this work, a new paradigm of learning called Active Learning is explored which uses user interaction to obtain annotations for a subset of the dataset. The goal of active learning is to achieve superior object detection performance with images that are annotated on demand. To realize active learning method, the trade-off between the effort to annotate (annotation cost) unlabelled data and the performance of object detection model is minimised.

Random Forests based method called Hough Forest is chosen as the object detection model and the annotation cost is calculated as the predicted false positive and false negative rate. The framework is successfully evaluated on two Computer Vision benchmark and two Carl Zeiss custom datasets. Also, an evaluation of RGB, HoG and Deep features for the task is presented.

Experimental results show that using Deep features with Hough Forest achieves the maximum performance. By employing Active Learning, it is demonstrated that performance comparable to the fully supervised setting can be achieved by annotating just 2.5% of the images. To this end, an annotation tool is developed for user interaction during Active Learning.

Keywords: Object Detection, Interactive Object Detection, Active Learning

Contents

List of Figures

List of Tables

Abbreviations

Acronym	**W**hat (it) **S**tands **F**or
MRI	**M**agnetic **R**esonance **I**maging
CAPTCHA	**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part
HoG	**H**istogram of **O**riented **G**radients
SVM	**S**upport **V**ector **M**achine
CNN	**C**onvolutional **N**eural **N**etwork
R-CNN	**R**egions with **C**onvolutional **N**eural **N**etwork features
YOLO	**Y**ou **O**nly **L**ook **O**nce
SSD	**S**ingle **S**hot MultiBox **D**etector
VGG	**V**isual **G**eometry **G**roup
WSOL	**W**eakly **S**upervised **O**bject **L**ocalization
CONV	**C**onvolutional layer
RGB	**R**ed-**G**reen-**B**lue
mAP	**m**ean **A**verage **P**recision
FIB-SEM	**F**ocussed **I**on **B**eam-**S**canning **E**lectron **M**icroscope
HeLa	**He**nrietta **La**cks

Chapter 1

Introduction

Artificial intelligence [2] is considered to drive the next industrial revolution [3] with far-reaching effects, like that of electricity [4]. This has led to immense speculation and demand to tap the economic potential of artificial intelligence in a variety of applications. As a result, there is a demand to extract actionable information from raw data captured from numerous applications like speech, healthcare, driving, consumer retail etc.

Data from such applications is captured by a variety of sensors that generate vast amounts of multimodal raw data [5, 6] i.e. images, video, speech, text etc. Among all modalities, the visual medium has dominated how living beings interact with their environment [7]. This has resulted in significant innovations in the image capturing technologies e.g. colour cameras, Kinect [5], HoloLens [8], MRI devices etc. Coupled with advances in manufacturing technologies and the Internet, cheap image sensors have been embedded in many devices e.g. mobile phones, laptops, medical devices, microscopes etc. and are easily shared. This incentivises efforts towards being able to extract actionable information and is actively researched by both academia [5, 9] and the industry [10–12]. Extracting actionable information from images is, therefore, the central focus of this thesis.

Automatically extracting information from images is addressed by computer vision which is a subfield of Artifical Intelligence. Such information at the objective level, grounds information within an image e.g. classification, segmentation, detection [9, 13, 14]. One of the fundamental problems of this category is to extract ”which” objects are present ”where” in a given image. Here, an ”object” is an entity contained by a 2D bounding box, ”which” classifies the bounding box into one of a predetermined set of class labels and ”where” localizes the bounding box within the image. This problem is termed as Object Detection [15] and is the focus of this thesis.

1.1 Motivation

Object detection is a fundamental building block for semantic image understanding and has been used in numerous applications such as tracking [16], pose estimation [17], image segmentation [18], virtual reality [8], autonomous driving [19], robot navigation [20] and biomedical image analysis [21].

The problem of object detection is tackled via representation learning [22]. In its simplest form, features extracted from each bounding box is classified into object / non-object using a model. The model is based on positive (object) and negative (non-object) training examples obtained through manual annotation [23] as shown in figure 1.2. There exist various techniques of model-learning based on the availability and granularity of manual annotations. In terms of annotation availability, learning is categorized as supervised [13, 24], semi-supervised [25, 26] and unsupervised [27, 28] which employ fully annotated to fully un-annotated training data.

State-of-the-art object detection techniques [13, 29, 30] typically depend on large fully annotated datasets [9, 31]. Therefore, fully and consistently annotated data is key to good object detection performance. In reality, however, obtaining ground truth annotations is a confusing, expensive and tedious task [32]. In real-world scenes, object detection is a difficult problem because there can be intra-class variations, view- point variations and articulation, illumination, background clutter, occlusion and motion blur as shown in figure 1.1. The variations due to articulation, pose and occlusion results in annotations biased on intra-/inter- human annotator. This is prevalent in medical imaging applications where a high degree of disagreement between annotation experts is common. Obtaining annotations also become prohibitively expensive based on availability, size and privacy restrictions of datasets and there is a need for expert annotators e.g. for medical applications.

In this regard, numerous approaches have been devised to ease the complexity of obtaining ground truth annotations. The most popular approaches are online annotation games [33] and CAPTCHA [34]. Annotations have also been obtained through crownsourcing [35] and through annotation firms [36, 37]. However, such solutions do not scale well with dataset size and do not address the issue of data privacy. A different approach is taken by Active Learning [32] which exploits the fact that examples are statistically dependent. Here, the annotation and model learning processes are interleaved. In each iteration, those examples, which are most diverse or beneficial wrt. the model, are annotated.

This thesis focuses on Active/Interactive Learning [1, 38–40] for object detection in images. The aim is to maximize object detection performance while minimizing annotation effort by iterating between model learning and annotating (beneficial) examples.

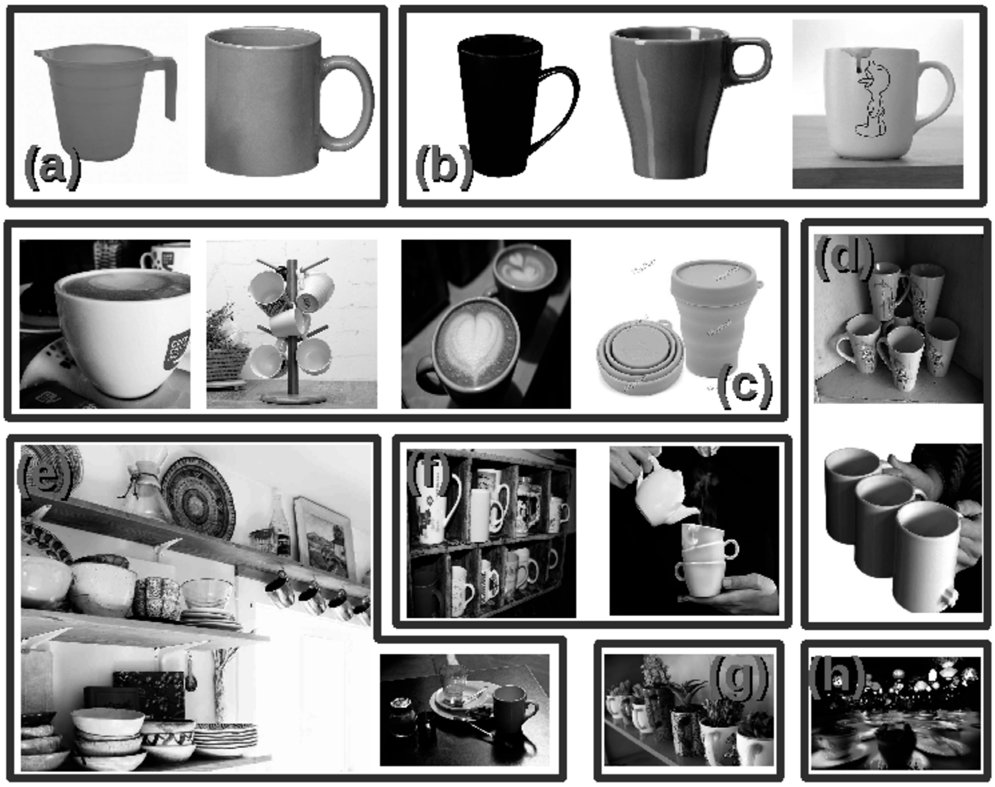


FIGURE 1.1: Why is object detection challenging? (a) discrete category labels merging bathing and coffee mugs together, (b) intra-class variations within coffee mugs, (c) viewpoint variations and articulation, (d) illumination, (e) background clutter, (f) occlusion, (g) alternative functionalities and (h) motion blur. Images courtesy of websites: magicemart, qualitylogoproducts, toxel, ikea, wayfair, foodspotting, alicdn, financialexpress, tinydeal, terapeak, designsponge, herpeculiarlife, thisiswhyimbroke, thisiswhyimbroke, wordpress, craftychica, netdna-cdn.

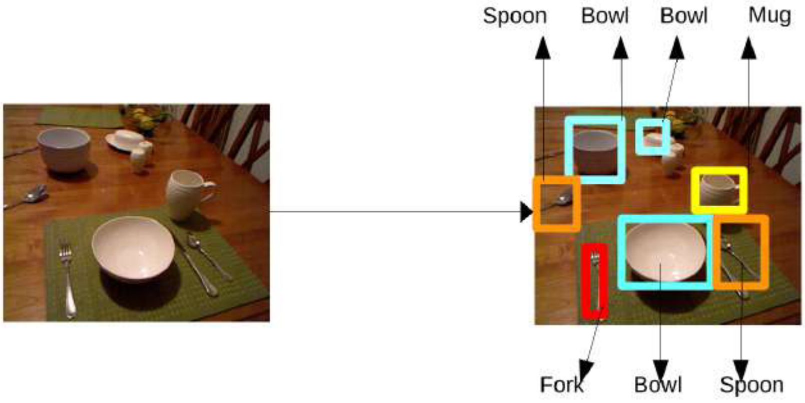


FIGURE 1.2: Bounding boxes of objects and their labels as annotations required for object detection problem is shown in (b) for the initial image (a).

1.2 Problem statement

To realise the goal of reduced annotation effort, a new paradigm of learning called Interactive Learning [1] is explored. In Interactive Learning, the learning model is updated continuously by querying human/oracle who verifies the annotation proposed by the model as illustrated in figure 1.3.

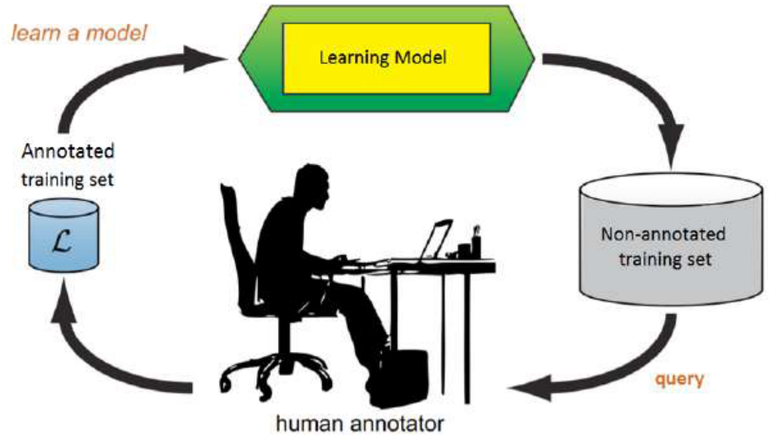


FIGURE 1.3: Interactive Learning cycle. Image adapted from [32]

This thesis explores two methods to realise Interactive Learning namely Incremental/-Passive/Random learning and Active Learning [1]. In Incremental Learning, a fixed set of examples are picked randomly from a pool of non-annotated data for verifying the outcome of the most recent model. The model is then updated based on the newly obtained annotations. A more optimal strategy might be to pick (statistically) independent examples, that are most diverse wrt. the most recent model. This method is called Active Learning where a fixed number of non-annotated examples most beneficial to the model are picked for annotation. The intuitive idea is to pick conditionally indpendent examples to maximize the diversity while minimizing the number of samples in the training dataset.

In this regard, *how to pick the most beneficial examples from the pool of non-annotated data?*

This is solved using a greedy procedure which iterates between minimizing the annotation cost and maximizing the accuracy of the object detection model. Minimizing the annotation cost amounts to selecting a subset of non-annotated examples to be presented for feedback from oracle/human, as illustrated in figure 1.4. Maximizing the accuracy of the model amounts to tuning the object detector to all annotated examples. In this regard, the problem context, its scope and contributions are presented are follows.

Problem Context

While the importance of Active Learning for object detection is presented in section 1.1, the following are important practical considerations:

- Real-time constraints: A fundamental requirement to realize an effective Active Learning framework is to choose a suitable model for object detection. In such scenarios, the model must have the capability of fast training/testing times. Also,

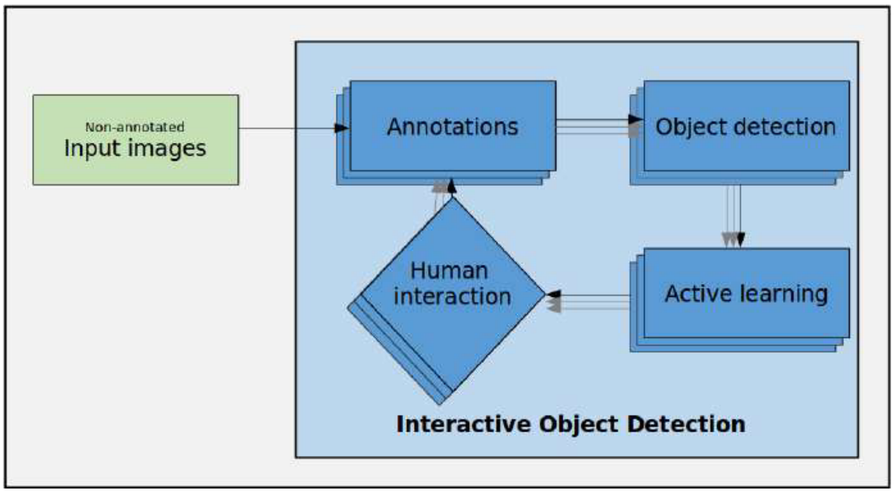


FIGURE 1.4: Figure describing the architecture of Interactive Object Detection with Active Learning. There is iteration between human interaction phase and object detection phase.

the model must be capable of learning from small/mid-sized datasets. In this regard, Hough Forests [30] is utilized as it has favourable runtimes, parallelizing potential and is shown to be robust to small/mid-sized datasets.

- Robust annotation cost: Another aspect of an effective Active Learning framework is to choose a robust annotation cost. The method [1] is adopted which is shown to be successful on real-world datasets and is inline with the Hough Forest model.
- Ease of verification: Finally, a simple user interface needs to incorporate verifications from oracle/human in an effort effective manner needs to be realized.

1.3 Scope of this work

In this work, the focus is on developing real-time interactive object detection pipeline using Hough Forests for object detection and Active Learning. Overall the following challenges are solved.

- The problem of object detection can be challenging because of factors like viewpoint variation, scale variation, deformation, occlusion, illumination conditions, background clutter and intra-class variation as shown in figure 1.1 . The next problem in object detection is solving two problems of locating objects and classifying objects at the same time using the same model.

  Both are solved using robust Hough Forests which uses deep features to defy viewpoint variation, scale variation, illumination conditions and intra-class variation. To solve both localization and classification problems, classification and regression nodes are incorporated in Hough Forests. Hough forests are explained in chapter 3

- The challenge in Active Learning is to find the most beneficial image to annotate, tuned to object detection problem.

  Beneficial images to annotate in each iteration are found using uncertainty sampling based Active Learning method [1]. From the experiments, it is proved that Active Learning performs better than choosing random images. The Active Learning method is explained in chapter 3 and experiments in chapter 5.

What is not in the scope of this work?

- Using deep learning methods for object detection: In this work real-time object detection is desired and deep learning based methods are slow to train.
- Multiclass detection is not incorporated in the Hough Forest-based object detector. Only one class of objects are detected per dataset.
- Comparision of different Active Learning query strategies (explained in section 2.3) are not shown in this work. Only uncertainty sampling based Active Learning method is used for Active Learning.

1.4 Contributions

The contributions of this thesis are as follows:

- An Active Learning [1] framework for object detection [30] is realized.
- Traditional HoG features are replaced by state-of-the-art Deep features for improved object detection (See section 4).
- A Web-based annotation tool to incorporate verification from orcale/human is implemented.
- Custom implementation of Hough forests completely in Python, in comparison with C++ based implementation. Visualizations obtained through the custom implementation are shown in figure 3.2. The code is available in the CD attached.
- Qualitative and quantitative results on four Carl Zeiss/benchmark datasets is presented.

1.5 Outline

The thesis is organised as follows:
Chapter 2 presents related work for object detection, Semi/Weakly- Supervision and

Active Learning.
Chapter 3 presents the methodology for Active Learning [1] and object detection [30].
Chapter 4 explains model parameters and presents implementation details.
Chapter 5 presents qualitative and quantative results on four (benchmark and Carl Zeiss) datasets.
Chapter 6 presents observations and conclusions along with future work.

Chapter 2

State of the Art

In this chapter, popular approaches to solve the problem of object detection are explained. Next, working with fewer annotated images, Interactive Learning and Active Learning are explained. State-of-the-art object detection methods, learning with partially annotated data and other Active Learning methods are explained to get a more relevant context.

Interactive Learning method in the context of computer vision problem domain is shown in figure 2.1. Object detection is a sub-field of Computer Vision. Depending on the annotation in the dataset, object detection can be solved using Supervised, Unsupervised, Weakly-supervised, Semi-supervised and Interactive Learning. In this work, Interactive Learning is focussed.

Combining the advantages of using deep learning and fast object detection based on Random Forests, method which uses deep features as an input to interactive object detector based on Hough Forests[1] is the main focus of this work.

2.1 Object Detection

The object detection algorithms combine the functional aspects of object location (where is the object?) [41] and object classification (which is the object?) [9] to detect objects.

2.1.1 Classical approaches

Earliest and most popular object detection method is using Haar features and cascade of classifiers [15] to detect faces. It uses Haar features [42] which are composed of five Haar templates. These features are extracted from a multi-scale sliding window. It uses Adaboost [43] for learning the cascade of classifiers. This was considered simple and

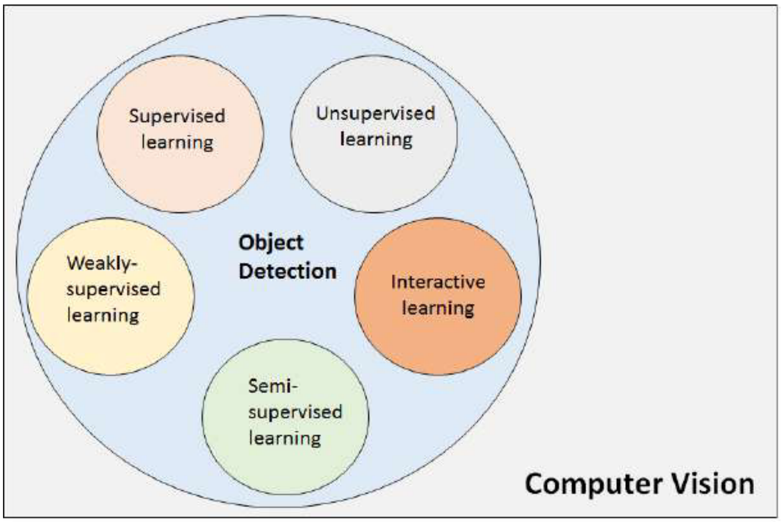


FIGURE 2.1: Interactive Learning method in the context of computer vision problem domain. Object detection is a sub-field of Computer Vision. Depending on the annotation in the dataset, object detection can be solved using Supervised, Unsupervised, Weakly-supervised, Semi-supervised and Interactive Learning. In this work, Interactive Learning is focussed.

fast by processing 15 frames per second and was used in point and shoot cameras which allowed real-time face detection.

Another method [44] uses Histogram Of Gradients features (HoG) and Support Vector Machines(SVM) [45] performed on a multiscale sliding window for person detection. HoG features were proved to be better than Haar features.

The next classical method to solve object detection problem with lesser parameters to tune in comparison to SVM is Random Forests [46]. The Random forest consists of an ensemble of decision trees arranged in a forest. Method [47] for face detection consists forests arranged in cascade structures influenced by method [15]. Another method Hough Forests [30] uses Hough Transform with Random Forests to detect generic shapes and object class instances. The input to the forest is HoG features extracted from images. Each tree in the forest consists of randomly placed classification and regression nodes together performing detection. At the leaves of trees, the detection is performed by weighted aggregation of evidence from regression. Hough forests are used as the object detection component of this work's Interactive Learning pipeline. This method is explained in detail in section 3.

Feature-based learning methods suffer from a problem that they only capture a small set of recognition cues and ignore other cues [48]. The features learned are image type specific and have to be re-learned for new image types. So there was a desire to develop algorithms to automatically learn features from images.

2.1.2 Convolutional Neural Networks

With improved computational power and large annotated datasets, the Convolutional Neural Network(CNN) paradigm today has largely been able to overcome the limitations of feature-based learning methods by automatically learning the features.

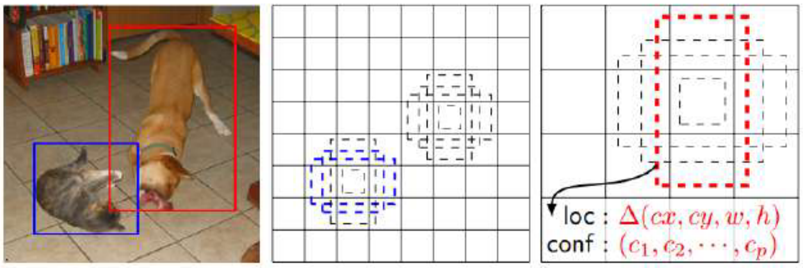
Regions with CNN features (R-CNN) [49] with CNN provided a 50% improvement over the best previous results on PASCAL VOC 2010 [50] dataset with feature-based methods. It follows a three-stage approach involving first generating object proposals using selective search [51] method and feature extraction using CNNs, lastly classification using SVMs. This method was slower because of CNN feature extraction applied to each proposal.

Quicker and modified approach Fast R-CNN [52] applied CNN to get features from the complete image instead of each proposal and then used both Region of Interest (RoI) pooling on the features with a final feed-forward network for detection. Usage of selective search slowed down the run-time of this method. In Faster R-CNN [53], a region proposal method was dropped to design a fully end-to-end trainable model. Also called Region Proposal network (RPN) it has a fully convolutional network which simultaneously predicts object bounding box and objectness scores for each position.

An alternative paradigm is explored in You Only Look Once (YOLO) [24]. Unlike other object detection methods which use an intermediate object proposals stage, the method employs a single neural network to simultaneously regress the spatial extent of bounding boxes and its class label. This is performed by dividing the input image into cell grids and the regressing bounding boxes and confidence scores for each cell. The system used has 24 convolutional layers, 2 fully connected layers followed by a final layer which predicts class probabilities and bounding box coordinates. This predictor model is applied to an image at multiple locations and scales.

The next method called Single Shot MultiBox Detector (SSD) [13] combined the functional aspects of Faster R-CNN and YOLO. Object localistion and classification are performed in a single pass of the CNN. It is built on VGG-16 [53] base architecture with addition of auxillary convolutional layers to extract features from multiple scales and aspect ratios to compute confidence and location loss for classification and regression tasks respectively. Figure 2.2 shows working of SSD with an example. This is among the state-of-the-art, real-time object recognition systems.

All the methods mentioned above are based on Fully Supervised Learning approach which requires a lot of annotated data to reach their full potential. Next, methods which work with partially annotated data are explored.


loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \cdots, c_p)$

(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

FIGURE 2.2: *SSD framework explained. a) and b) shows auxillary convolutional layers with different scales (e.g. 8 × 8 and 4 × 4). For each auxillary convolutional layer, both the shape offsets and the confidences for all object categories (c_1, c_2, $\cdots$, c_p) are predicted. Auxillary convolutional layers are matched to the ground truth boxes. For example, the two auxillary convolutional layers with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum of localization loss and confidence loss.* Caption adapted from [13]

2.2 Learning with partially annotated data

2.2.1 Weak annotations

Given the difficulty in acquiring annotations, data annotation is a research area in itself. There are methods which make the job of annotations easier by introducing simplified annotation methods. In extreme clicking method [54], the bounding box annotations are provided in form of four points ($x_{lefttop}$,$y_{lefttop}$), ($x_{righttop}$,$y_{righttop}$), ($x_{rightbottom}$,$y_{rightbottom}$),($x_{leftbottom}$,$y_{leftbottom}$). This speeds up the process of annotation by 3x. In another method, object bounding-boxes are derived from eye-tracking [55] by fixations from eye movement of the annotators. In [56], detection is treated as a multi-stage problem. In the first stage, the user clicks on the centre of the object for annotating a bounding box. These annotations are used by Weakly-supervised object locations to localize the objects. These localised bounding boxes are used in next stage for detection. In semantic segmentation [57], annotating each pixel is a time-consuming task. Pixel-wise annotations are replaced by scribbles which is a user-friendly way of annotation.

Next, learning methods which work with partially annotated data are explored.

Weakly-Supervised Learning

Weakly-Supervised Learning [26, 58] makes use of image or instance level annotation for learning. This method is gaining popularity due to easily available image tags on images or videos, which can serve as image level annotations. With respect to object detection, the learning task is to localize and recognise the object using only image/instance level annotation but not the location of the object. This task is often termed as Weakly-supervised object localization (WSOL) [59]. The recent works in WSOL involving using

CNN to extract features improved the detection performance but it still slow compared to Fully Supervised Learning methods.

Semi-Supervised Learning
Semi-Supervised Learning [60, 61] makes use of small amount of annotated data with a large amount of unlabelled data. This has an underlying assumption of data being arranged in a cluster or along low dimensional manifolds. In self-learning method [62], the annotated data act as the base for object detector. The detections are propagated to unlabelled data and are further used to train the detector. There are chances that it becomes confidently wrong. In graph-based learning [63], the annotated data propagates the annotations to unlabelled data with a lower confidence. This is an iterative process which can go infinitely and can be difficult to scale. Max-margin and ensemble methods [64] develop a decision boundary based on the density of the mixture of annotated and unlabelled examples. These methods are usually tied to SVM based methods.

2.2.2 Transfer Learning

Due to the advent of CNNs, the knowledge (network weights) acquired from training very large datasets are transferred to learning unseen but related datasets. This transfer of knowledge is called transfer learning [65, 66]. This transfer could be done by using CNN as feature extractor or fine-tuning to already trained CNNs. This is, however, CNN method dependent and cannot be used with other methods.

2.2.3 Reinforcement Learning

Reinforcement Learning is learning policy based on reward or punishment. In the context of learning with partially annotated data, a policy for image annotation is learned instead of image analysis. In work [67] the agent is rewarded for actions that reduce the uncertainty about the unobserved images based on recurrant neural network to perform active completion of panoramic natural scenes and 3D object shapes. For the kind of data, setup and annotations present, this method cannot be used in our work due to the reason that the method is noise sensitive.

2.3 Interactive Learning

Interactive Learning is performed by a collaboration between human and machine for the process of learning. The interactive component can come from providing feedback by correcting annotations or providing new annotations, in each iteration. The user verification can be obtained for all images or a fixed number of images or for ”beneficial” images.

In [59] annotations for bounding box are verified during each iteration produced automatically by object detection method. After verification, re-localization of object and re-training of the detector are performed for every iteration. Here the verification is sought for all images.

Incremental learning

In incremental learning [1], a fixed set of images for eg. every n^{th} image or randomly chosen image is annotated. These images are shown to the user via an interface to correct the annotations. In each iteration, an optimal threshold is computed to differentiate positive images from negative images. The order in which the images are annotated influences the detector's performance. Therefore, in next method called active learning, the images are selected in descending order of their difficulty to annotate. The comparison between Incremental and Active Learning is shown in figure 2.3

Active learning

A comprehensive literature survey related to Active learning is provided in [38–40]. Active Learning requires querying most beneficial images from the pool of data. Following are query methods :

- **Random sampling**

  Also called Incremental Learning, in this method, a random image is chosen in every iteration instead of using any strategies. This strategy works best when the dataset contains similar images. But it can lead to learning of non-beneficial, repetitive images. The effectiveness of learning beneficial over random sampling is illustrated in figure 2.3. Random sampling is used as a baseline method in experiments section 5.

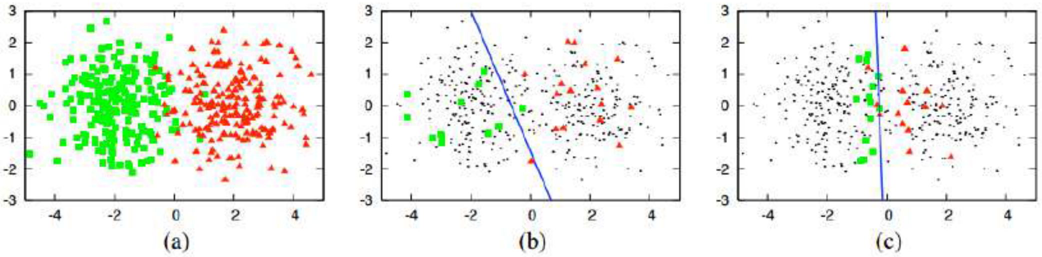


FIGURE 2.3: *An illustrative example of the difference between Random and Active Learning. (a) A toy dataset of 400 instances, evenly sampled from two class Gaussians. The instances are represented as points in a 2D feature space. (b) A logistic regression model trained with 30 annotated instances by picking random instances. The line represents the decision boundary of the classifier (70% accuracy). (c) A logistic regression model trained with 30 beneficial instances.* Image and caption from [32].

- **Query-By-Committee**

  This is an Active Learning method which queries form pool of unlabelled data. In each iteration, a committee of learning methods from the current training set is chosen. All the learning methods predict the output for each unlabelled image in the dataset. The image whose prediction highly differs among the methods in the committee is selected to be annotated. The committee can be formed by same learning method with varying parameters [68] or by using ensemble learning [69]. Disagreement of the committee can be measured using vote entropy or KL divergence [70].

- **Expected Model Change**
  In this method, the instance that results in the greatest change from the current model is selected for annotation in each iteration. One of the famous works using this method is [71] where, for each example of an unlabelled set, the expected change of model predictions is calculated and marginalized over the unknown label. This results in a score for each unlabeled example that can be used for Active Learning with a Gaussian process for classification. This method could be inefficient if both the feature space and set of labels are very large.

- **Expected Error Reduction**
  In this method, the beneficial image is not selected on the basis of how the model is likely to change, but how the generalization error is reduced. The unlabelled pool of data is called validation set. If there is a highest decrease in generalization error on the validation set when an image is not used, this image is considered to be the beneficial image. In [72] a regressor is trained that predicts the expected error reduction for a data in a particular learning state. This method is computationally expensive because it requires estimating expected future error for each query and a new model must be incrementally re-trained for each query labelling which iterates over entire pool.

- **Uncertainty Sampling**
  This Active Learning method queries an image whose annotation is most uncertain. The uncertainty can be decided on by the confidence obtained from score predicted by object detector [1] or entropy centred objective function as in [73]. It works on the concept that the image patches generated from the same object share the same label and differ slightly in their relative distance from the centre of the object. These patches will also have the same prediction from the learning method. The images having higher entropy generally indicate high variability among its patches. These images with higher entropy can contribute to increasing the performance of the learning method. The learning method used in this case is CNN. The Active Learning method used in this thesis [1] is based on uncertainty sampling method. It is explained in section 3. The disadvantage of using uncertainty based query strategy is that there is a possibility of learning noisy images because these kind of images have the highest uncertainty.

Further, deep learning is incorporated into Active Learning called ”Deep Active Learning” [73] based on uncertainty sampling and CNNs. In [74] Bayesian Deep Learning is used with high dimensional data to perform classification task. These methods suffer from slow training process because of CNNs used as the learning method.

This work is based on Interactive Object Detection [1], where a Hough Forest-based object detector which is optimised based on a realistic annotation cost model. The input to the object detector is HoG features obtained from images. The detailed explanation of this method is provided in Chapter 3. Figure 2.4 shows example output from Interactive Object Detection method from three datasets with images and with annotation cost. This work shows results on real-world datasets with fast object detection in order to perform interactive detection with user feedback.

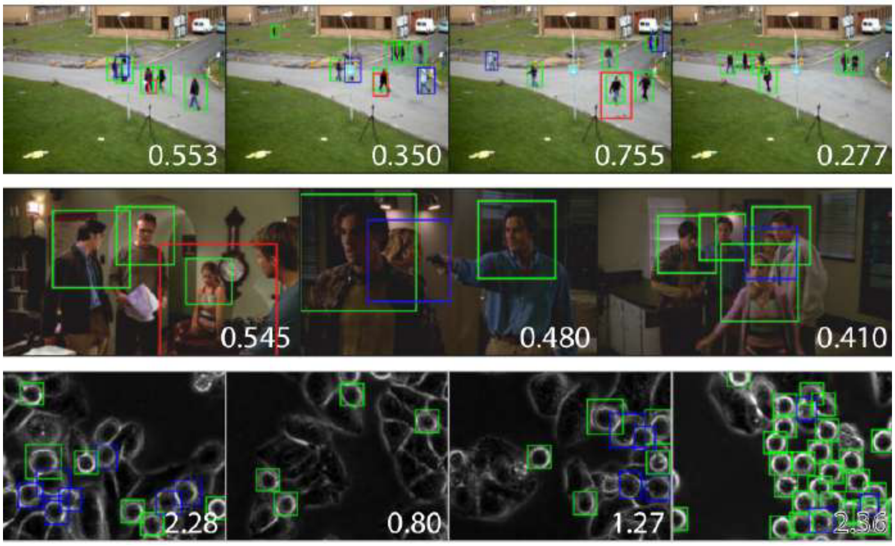


Figure 2.4: *Results of the Interactive Object Detection framework for annotation from work [1]. Green, red, and blue bounding boxes denote true positives, false positives, and false negatives respectively according to the annotation task. The numbers denote the predicted annotation cost, i.e., the cost to correct all detection errors in an image. The predicted annotation cost allows the user to select images for correcting detections and updating the object detector.* Caption adapted from [1].

Chapter 3

Technical background

3.1 Hough Forests for object detection

Hough transform

Hough transform [1, 75, 76] can be used to find occurrences of a particular shape in images. The requirement is for the shape to be represented in parametric form. It was basically designed to find analytically defined shapes such as lines, circles and ellipsoids. For instance, here Hough Transform to detect lines can be shown in figure 3.1. In image space, a line can be expressed in cartesian co-ordinates with parameters ρ, θ where ρ is the length of normal from the origin to the line to be found, and θ is the orientation of ρ with respect to the x-axis as: $\rho = x \cos\theta + y \sin\theta$. For a given ($x_i$ and y_i) and (x_j and y_j) the family of lines that pass through this point are represented by: $\rho = x_0 \cos\theta + y_0 \sin\theta$ as shown in figure 3.1 (a). Here, the observed variables are from $\mathbb{R} \in (x_j, y_j)$ and parametric space is $\mathbb{R} \in (\rho, \theta)$

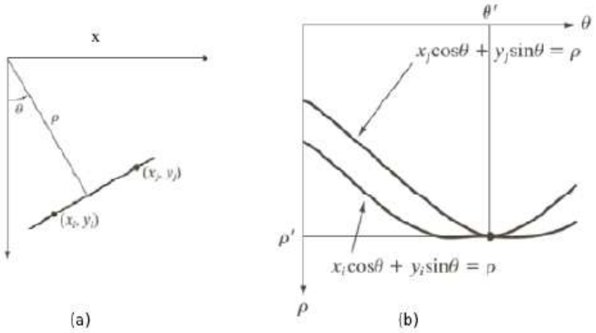

x
y
θ
ρ
(x_j, y_j)
(x_i, y_i)
(a)
θ'
θ
$x_j \cos\theta + y_j \sin\theta = \rho$
ρ'
$x_i \cos\theta + y_i \sin\theta = \rho$
ρ
(b)

FIGURE 3.1: Simple example of Hough Transform. A line shown in (a) is transformed into curve shown in (b).

The possible values of (ρ,θ) are plotted against specific (x_i and y_i) and (x_j and y_j) , values in cartesian image space, to curves in polar Hough parametric space. This transformation is called Hough Transformation for lines. This allows for efficient and robust implementation which is occlusion, noise and deformation independent.

The observation is that locally observed variables (x_j, y_j) cast a vote for global confgurations (ρ, θ).

This transformation is quantizing Hough parameter space into finite bins. In this Hough parametric space, points on a line which share parameters, intersect at a common point (ρ,θ). Each value (x_i,y_i) and $(x_j\ ,\ y_j)$ is transformed into discretised (ρ,θ) curve and all the bins along this curve are aggregated as shown in figure 3.1 (b). The peaks in Hough space bins represent the presence of a line. As there could be multiple possible aggregated bins, the useful peaks are found by non-maximal suppression [77]. Hough transform is extended to detecting other shapes which are analytically defined shapes and object class instances [30][78].

Generalised Hough Transform

Hough transform can be applied for detection of any shape based on aggregation of evidence of the presence of the transformed curve with respect to the image of interest [30]. These evidences are called "Hough votes" The parametric space here is called "Hough space". The parameters here can be set of object location in image, scales, aspect ratios etc. The detection step is finding peaks of aggregation of Hough votes in Hough space by non-maximal suppression. Next, how Hough Transform is used with Random Forests to detect objects is presented.

Random forests with generalised Hough Transform

The previous research work is re-iterated here to explain random forests [29]. Hough forests are a group of decision trees.

Random forest [46] is an ensemble of decision trees [79]. Decision trees are the hierarchical tree-like representations consisting nodes and branches. During decision tree learning, it looks into attributes in data to split the data into subsets. Splitting is done according to different criteria like Gini impurity, regression or entropy loss. Here binary tests are used to determine the best splits. The splitting is continued until the subsets are pure and the results are stored in the leaf node. One disadvantage of decision trees is it can easily over-fit the data. Random forests, on the other hand, is an ensemble of decision trees whose input is a random subset of the actual data. The final result of the random forest is the mean or mode of the result of individual decision trees.

The following equations are used as is and explanations are adapted from work [1, 30, 80].

In the Hough Forests implementation used from [30, 80], the features from the image are mapped to corresponding votes in Hough space. y denotes the mid-point of input image patch with dimension D. The features are denoted by $(I^1(y)$,$I^2(y)$,....$I^f(y))$ where F denotes the number of feature channels. $\mathcal{I}$ denotes mapping from the patches to features. In work [30], this mapping is done by HoG and RGB features. $\mathcal{H}$ denotes Hough space and $\mathbf{h}$ denotes hypothesis.

The leaves of the trees $\{L\}$ model the mapping from the patch with centre y to probabilistic Hough vote denoted by:

$$\mathcal{L} : (y, \mathcal{I}) \rightarrow p(h|L(y)) \tag{3.1}$$

$p(h|L(y)$ is the distribution of Hough votes in Hough space. Learning mapping $\mathcal{L}$ is also learning Hough Forest model which is explained in next section.

3.1.1 Training

The trees are trained on image patches. For each class, $c \in C$ training images are available. Positive image patches are sampled from inside bounding box and negative patches are sampled from the background as shown in figure 4.1. Each patch contains information regarding its class. In addition, each positive patch also contains information about offset from the centre of the patch to the centre of the object. The D-dimensional image is used to build Hough tree T and forest $\mathcal{T} = \{T_t\}$ from the set of patches $\{\mathcal{P}_i = (I_i, c_i, \mathbf{d}_i)\}$ where $\mathbf{d}_i$ is offset vector.

The tree consists of leaf and test nodes. Each non-leaf node is assigned a binary test whose domain are the features $\mathcal{I}_{\rangle} = (I_i^1, I_i^2....I_i^F)$. A binary test $t_\theta(I_i)$ on a patch is parametrised by $\theta = \{\ f,\ \mathbf{p},\ \mathbf{q},\ \tau\ \}$ where $f \in\ 1, 2, 3, ..., F$ is generated on two randomly chosen positions $\mathbf{p} \in \mathbb{R}^D$ and $\mathbf{q} \in \mathbb{R}^D$ within feature matrix and a real-valued threshold τ. The test is:

$$t_\theta(I_i) = \begin{cases} 0 & if I_i^f(\mathbf{p}) < I_i^f(\mathbf{q}) + \tau, \\ 1 & otherwise \end{cases} \tag{3.2}$$

The evaluation criteria for the binary test is to minimise the uncertainty for either discrete or continuous random variables which are class labels c and offset vectors $\mathbf{d}$.

Each non-leaf node in the tree is either classification (to minimise the uncertainty of class label) or a regression node (to minimise the uncertainty of offset vectors) decided randomly. Let P be set of incoming patches. For a classification node, the measure of the uncertainty of set of patches is given by:

$$U_1(P) = -\,|P| \sum_{c \in C} p(c|P) \ln(p(c|P)) \tag{3.3}$$

where $|P|$ is the number of patches in set P and $p(c|P)$ is the proportion of patches with label c in P. Minimising this expression for a node corresponds to maximising the information gain.

For regression node, the measure of uncertainty of offset vectors is given by:

$$U_2(P) = \sum_{c \in C} (\sum_{d \in D_c^P} \left\| \mathbf{d} - \tfrac{1}{|D_c^P|} \textstyle\sum_{d' \in D_c^P} \mathbf{d}' \right\|)^2 \tag{3.4}$$

There are no displacement vectors $\mathbf{d}$ for negative patches. D_p^c is the set of all offsets of patches from class c in set P.

At each node, a pool of binary tests $\{ \ \theta^k \ \}$ are generated with random values of f, $\mathbf{p}$, $\mathbf{q}$ and τ sampled uniformly. Binary test θ for a test node is chosen in a greedy fashion from a set $\{\theta^k\}$. All the patches arriving at a node are evaluated for the set of binary tests θ^k and the best binary set is chosen and stored at the node. Best test should minimise the following minimization objective:

$$\underset{k}{argmin}(U_*(\{P_i|\theta^k = 0\}) + U_*(\{P_i|\theta^k = 1\})) \tag{3.5}$$

where $*$ indicates uncertainty measure for classification (U_1) or regression (U_2).

To construct a leaf node L , information from incoming patches are used to store the class probability $p(c|L)$ and a list of offset vectors $D_c^L = \{\mathbf{d}_i\}_{ci=c}$.

The tree is constructed recursively starting from the root by choosing a binary test and children nodes are constructed until a stopping criterion is reached. The stopping criteria could be the tree reaching its maximum depth (in our case 15 trees) or the number of patches falling below a threshold (in our case 2). When one of this criterion is met, leaf nodes are constructed with information of class probabilities $p(c|L)$ and list of offset vectors $D = \{\mathbf{d}_i\}_{c_i=c}$. One example of a trained Hough Forest with depth 3 can be seen in figure 3.2 where the image patches are fed into Hough Forests and they traverse in the forest splitting in regression or classification node to reach leaf where offset vectors and the probability of positive patches are stored. In the figure, the positive and negative patches are fed into the root of the tree. The first node is a classification node which differentiates negative patches from positive patches. The right child is also a classification node. The left child is a regression node which differentiates front part of the horse and back part of the horse. At the leaf nodes, as shown in the last row, the class probability $p(c|L)$ and a list of offset vectors $D_c^L = \{\mathbf{d}_i\}_{c_i=c}$ centered at the red point are stored.

3.1.2 Testing

During detection phase as shown in figure 3.3, the test image patches pass through each tree in Hough forest. The leaves that the patches arrive cast votes in Hough space $\mathcal{H}$. The Hough space is composed of parameters image scale and position.

Let a patch with the centre at position y be $P_y = (I_y, c_y, \boldsymbol{d}_{c(y)}) \in \Omega \subseteq R^D$ in the test image. Here Ω is set of all pixel locations, I_y is the set of observed features of the patch,

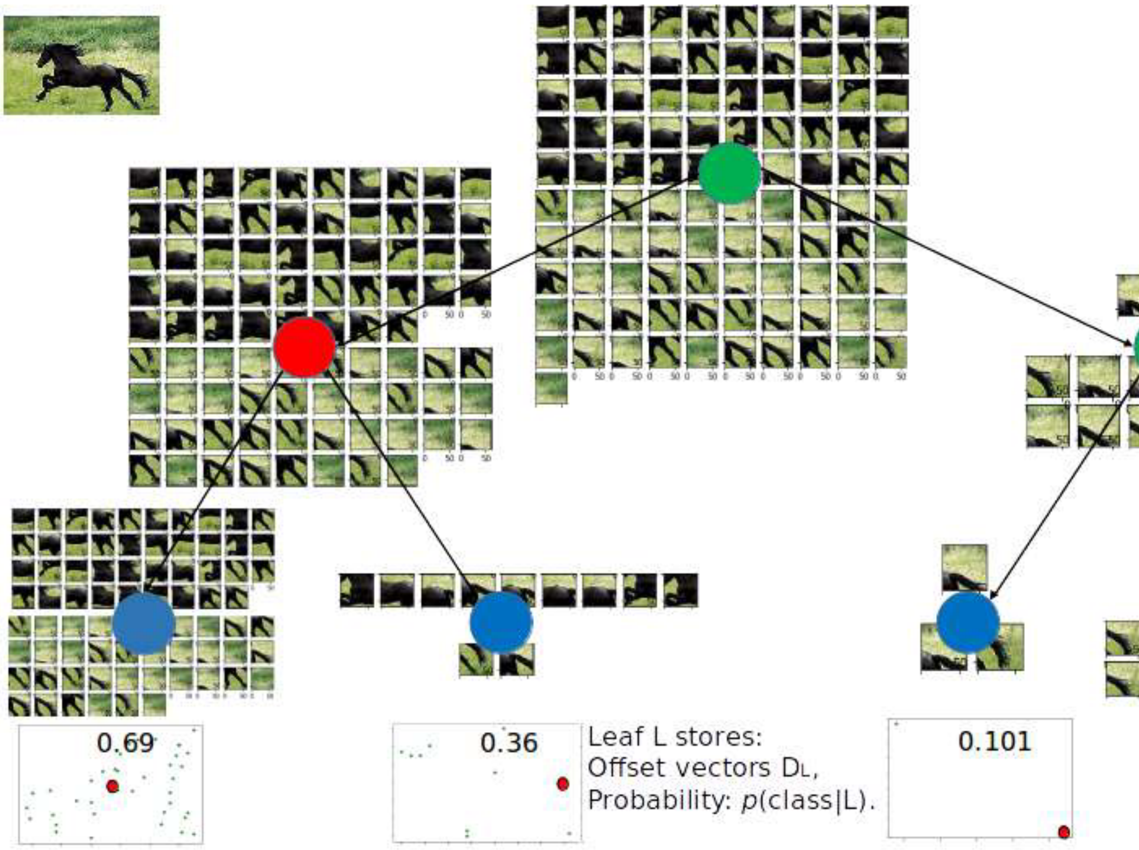


FIGURE 3.2: Example of trained Hough Forest. The positive and negative patches are fed into the root of the tr classification node which differentiates negative patches from positive patches. The right child (in green) is also a is a regression node (in red) which differentiates front part of the horse and back part of the horse. At the leaf nod probability $p(c|L)$ and a list of offset vectors $D_c^L = \{\mathbf{d}_i\}_{c_i=c}$ centered at the red point ar

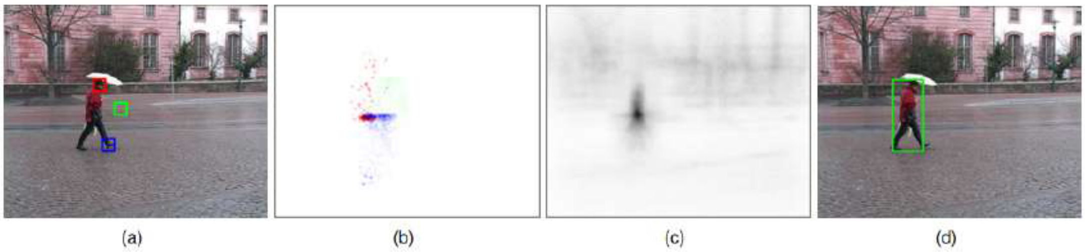


FIGURE 3.3: *For each of the three patches emphasized in (a), the pedestrian class-specific Hough forest casts weighted votes about the possible location of a pedestrian (b) (each colour channel corresponds to the vote of a sample patch). Note the weakness of the vote from the background patch (green). After the votes from all patches are aggregated into a Hough space (c), the pedestrian can be detected (d) as a peak in this image.* Caption and image from [1]

c_y is the hidden class label and $\boldsymbol{d}_{c(y)}$ is the hidden displacement from the patch to the unknown object's centre. Based on the feature I_y , patch P_y ends in a leaf $L(y)$. Let $\boldsymbol{h}$ be the hypothesis for the object belonging to class c with size s and centered at $x \in \Omega$. The conditional probability $p(\boldsymbol{h}(c, x, s)|L(y))$ can be computed as

$$\begin{aligned} p(\mathbf{h}(c,\mathbf{x},s)|L(y)) &= \sum_{l\in C} p(\mathbf{h}(c,\mathbf{x},s)|c(\mathbf{y})=l, L(\mathbf{y}))\cdot(c(\mathbf{y})=l|L(\mathbf{y})), \\ &= p(\mathbf{h}(c,\mathbf{x},s)|c(\mathbf{y})=c, L(\mathbf{y}))\cdot(c(\mathbf{y})=c|L(\mathbf{y})), \\ &= p\left(x = y - \frac{s}{s_u}\mathbf{d}(c)|c(\mathbf{y})=c, L(\mathbf{y})\right)\cdot p(c(\mathbf{y})=c|L(y)), \end{aligned} \tag{3.6}$$

where s_u is the unit size of the training data. $p(c|L)$ is estimated as the proportion of patches per class label reaching the leaf after training, the distribution $p(\boldsymbol{h}(c, x, s)|c(y) = c, L(y))$ can be approximated by a sum of Dirac measures δ_d for the displacement vectors $\boldsymbol{d} \in D_c^L$:

$$p(\mathbf{h}(c,\mathbf{x},s)|L(y)) = \frac{p(c(y)) = c|L(y)}{\left|D_c^{L(y)}\right|}\left(\sum_{d\in\left|D_c^{L(y)}\right|}\delta_d\left(\frac{s_u(y-x)}{s}\right)\right) \tag{3.7}$$

For the entire forest $\mathcal{T}$, features of the patch are passed through all the trained trees and average the probabilities from equation 3.7 from different leaves:

$$p(h|I_y) = \frac{1}{T}\sum_{t=1}^{\left|\mathcal{T}\right|} p(h|L_t(y)), \tag{3.8}$$

where $L_t(y)$ is the corresponding leaf for tree T_t. The votes from all patches of the image are accumulated in the Hough space $\mathcal{H}$:

$$p(h|I_y) = \sum_{y \in \Omega} p(h|I_y) \tag{3.9}$$

The modes of $p(\boldsymbol{h}|I)$ can be obtained by searching for local maxima using a Parzen estimator with a Gaussian kernel K:

$$\begin{aligned} \hat{p}(\boldsymbol{h}|I) &= \sum_{\boldsymbol{h}' \in \boldsymbol{h}} w_h' \cdot K(\boldsymbol{h} - \boldsymbol{h}'), where \\ w_h' &= \sum_{y \in \Omega} \sum_{t=1}^{|\mathcal{T}|} \sum_{d \in D_c^{t(y)}} \frac{p(c(y) = c|L_t(y))}{\left|D_c^{L_t(y)}\right|} \delta_d \left(\frac{s_u(y-x)}{s} \right) \end{aligned} \tag{3.10}$$

The weight of a hypothesis w_h' accumulates votes that support similar hypotheses $\boldsymbol{h}'(c, x, s)$ $\in \mathcal{H}$. After all votes are cast, $\hat{p}(\boldsymbol{h}|I)$ represents the sum of the weights of the hypotheses in the neighbourhood of $\boldsymbol{h}$ weighted by a Gaussian kernel K. While the location of a local maximum $\hat{\boldsymbol{h}}(c, x, s)$ encodes class, position and size of the object, the value of $\hat{p}(\hat{\boldsymbol{h}}|I)$ is not a probability but serves as a confidence measure for each hypothesis.

Example detection from the experiments conducted is shown in figure 3.4. Figure 3.4 (a) shows original image, with its Hough space and next detection by non-maximal suppression. And figure 3.4 (b) shows Hough space obtained for different scale which detects the objects having different scales, the first Hough space with scale=1 detects smaller objects and second Hough space with scale=0.5 detects bigger objects.

3.2 Active learning

Threshold estimation

In this section, implementation of Active Learning along with Hough Forests is described. Hough forest provides a score for each bounding box (hypothesis) detected. These scores can be thresholded to accept or reject the bounding box.

During Hough Forest detection, the positive and negative hypothesis needs to be distinguished using the bounding box scores. The detection scores corresponding to positive hypothesis are denoted by S_{pos} and scores from negative hypothesis denoted by S_{neg}. The distribution of scores conditional to a positive or negative hypothesis is denoted by $p(s|pos)$ and $p(s|neg)$. These conditional probabilities are modelled well by Gamma distributions [81]:

$$p(s|pos) = \gamma(k_{pos}, \theta_{pos}) p(s|neg) = \gamma(k_{neg}, \theta_{neg}) \tag{3.11}$$

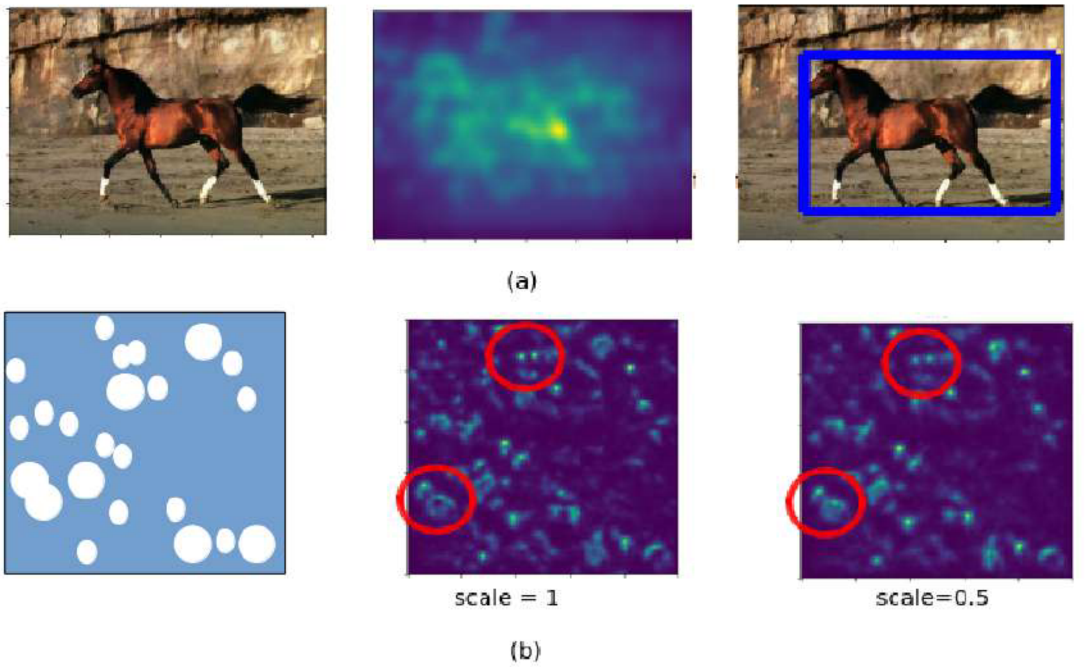


Figure 3.4: Testing part of Hough Forests explained with an example. (a) The Original image, with its Hough space and next detection by non-maximal suppression. (b) Synthetic dataset image with Hough space obtained for different scale detects the objects having different scales, the first Hough space with scale=1 detects smaller objects and second Hough space with scale=0.5 detects bigger objects

k and θ are the parameters of the gamma distributions obtained from S_{pos} and S_{neg}. The optimal threshold is formulated as that value of threshold τ which minimises the probability of false positive FP and false negative FN hypothesis for a given τ. This can be mathematically written as:

$$\underset{\tau}{argmin} p(FP|\tau) + p(FN|\tau) \tag{3.12}$$

based on above equations, following equations can be formulated:

$$p(FP|\tau) = p(neg) \int_{\tau}^{\infty} p(s|neg)ds \tag{3.13}$$

$$p(FN|\tau) = p(pos) \int_{0}^{\tau} p(s|pos)ds \tag{3.14}$$

where $p(pos) = \frac{|S_{pos}|}{|S_{pos}|+|S_{neg}|}$ and $p(neg) = 1 - p(pos)$. The initial threshold is calculated to be median of all the positive and negative hypothesis scores.

For every new score, the parameters of gamma distribution are updated. Initial threshold estimation requires at-least two annotated images. With more training examples, a better threshold can be estimated. The initial threshold is higher and converges after few iterations as shown in figure 3.6.

In figure 3.5 which shows gamma probability density function(PDF) of the scores, the intersection of two PDFs shows the scores which are most uncertain ie. false positive scores (red hashed regions) and false negative scores (blue hashed regions). The score which clearly distinguishes the false positive scores from false negative scores is considered optimal threshold (green line).

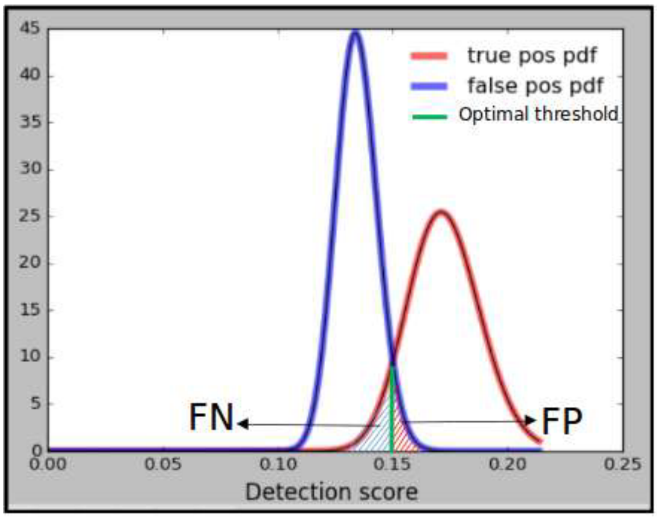


FIGURE 3.5: Detection scores of positive (blue) and negative (red) detection scores modelled by the gamma distribution. These distribution parameters are updated after each incremental training step. false positive scores (red hashed regions) and false negative scores (blue hashed regions). The score which clearly distinguishes the false positive scores from false negative scores are considered optimal threshold (green line)

Annotation cost

As the order of annotating images is responsible for the quality of detections, next Active Learning way of interactive detection is explored. Here the most beneficial images which have highest annotation cost is selected to annotate first. The annotation cost is formulated as:

$$f_{pred}(S,\tau) = \sum_{s \in S} p(FP|s,\tau) + p(FN|s,\tau) \tag{3.15}$$

and

$$p(E|s,\tau) = \frac{p(s|E,\tau)p(E|\tau)}{p(s|neg)p(neg) + p(s|pos)p(pos)} \tag{3.16}$$

where $E \in FP, FN$ and

$$p(s|FN,\tau) = \begin{cases} 0 & ifs \geq \tau \\ \frac{p(s|pos)}{\int_0^{\tau} p(s|pos)ds} & ifs < \tau \end{cases} \tag{3.17}$$

$$p(s|FP,\tau) = \begin{cases} \frac{p(s|neg)}{\int_{\tau}^{\infty} p(s|neg)ds} & ifs \geq \tau \\ 0 & ifs < \tau \end{cases} \tag{3.18}$$

The intuitive understanding of this equations can be explained using figure 3.5. The hashed region in the figure comprises the scores which are most uncertain. The equation 3.16 finds the image for which the area of uncertainty of all its scores is very high.

Thus, during every iteration of Active Learning, an image with highest annotation cost is found. For this image, the annotation from the user is obtained and the iteration is continued until the stopping criterion is reached. The stopping criterion is when the annotation cost is 0 as shown in figure 3.6.

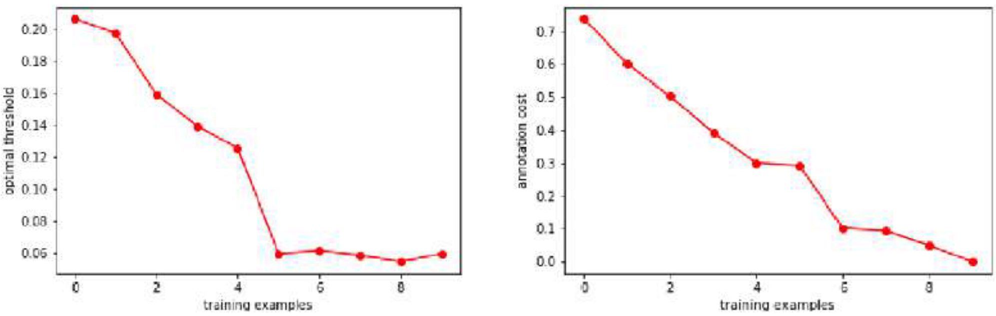


Figure 3.6: Convergence of (a) optimal threshold and (b) annotation cost over 10 iterations

The implementation of Active Learning is shown in the algorithm:

Algorithm 1 Active learning algorithm

```
procedure getBeneficialImage($images$)
    $\tau \leftarrow median(prevsScores)$
    $S \leftarrow houghForestPredict(images)$
    $maxAnnotationCost \leftarrow 0$
    for s in S do
        if $s \geq \tau$ then
            $S_{pos} = s$
        else
            $S_{neg} = s$
    $p(pos) \leftarrow \frac{|S_{pos}|}{|S_{pos}|+|S_{neg}|}$
    $p(neg) = 1 - p(pos)$
    $k_{pos}, \theta_{pos} = getGammaParam(S_{pos})$
    $k_{neg}, \theta_{neg} = getGammaParam(S_{neg})$
    $min\tau = min(S)$
    $max\tau = max(S)$
    $\tau_{list} = uniform(min\tau, max\tau, 100)$
    for $\tau$ in $\tau_{list}$ do
        $\int_{\tau}^{\infty} p(s|neg)ds = gammaCdf(t, k_{pos}, \theta_{pos})$
        $\int_{0}^{\tau} p(s|pos)ds = gammaCdf(t, k_{neg}, \theta_{neg})$
        $p(FP|\tau) = p(neg) \int_{\tau}^{\infty} p(s|neg)ds$
        $p(FN|\tau) = p(pos) \int_{0}^{\tau} p(s|pos)ds$
    for $S, image\ \ in\ \ images$ do
        $annotationCost = 0$
        for $s\ \ in\ \ S$ do
            if $s \geq \tau$ then
                $p(s|FN, \tau) = 0$
                $p(s|FP, \tau) = \frac{p(FP|\tau)p(neg)}{\int_{\tau}^{\infty} p(s|neg)ds}$
            else
                $p(s|FP, \tau) = 0$
                $p(s|FP, \tau) = \frac{p(FN|\tau)p(pos)}{\int_{0}^{\tau} p(s|pos)ds}$
            $p(FP|s, \tau) = \frac{p(s|FP,\tau)p(FP|\tau)}{p(s|neg)p(neg)+p(s|pos)p(pos)}$
            $p(FN|s, \tau) = \frac{p(s|FN,\tau)p(FN|\tau)}{p(s|neg)p(neg)+p(s|pos)p(pos)}$
            $annotationCost+ = p(FP|s, \tau) + p(FN|s, \tau)$
            if $annotationCost \geq maxAnnotationCost$ then
                $maxAnnotationCost = annotationCost$
                $beneficialImage = image$
    return $beneficialImage$
```

Chapter 4

Implementation details

This chapter presents implementation details of the Hough Forests, the Active Learning framework and the web-based annotation tool used to gather annotation from oracle/human.

4.1 Hough Forest

As discussed in section 3, Hough Forests uses generalized Hough Transform for generic shapes/objects. The details of its components are described in following sections.

4.1.1 Patch extraction

The input to all experiments is RGB images and ground truth bounding boxes of positive examples. In order to simplify the training procedure, positive bounding boxes are rescaled to its median value. Testing is performed in scale space, corresponding to the modes of scale distribution in the training data.

Patch extraction

The input to Hough Forests are patches of fixed $n \times n$ size patches extracted from the training images. The size of the patches is fixed as 16×16. Positive and negative patches are randomly sampled from within and outside the (scaled) positive examples respectively. Instead of using just the RGB image patches, a simplified image represen-

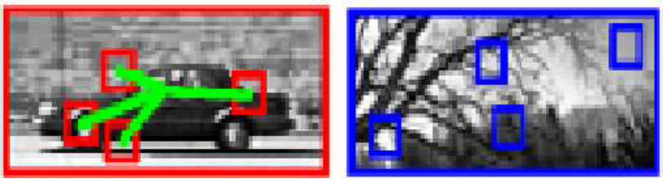


FIGURE 4.1: Positive patches extracted from inside bounding box and negative patches extracted the from the background.

tation which consists of ”useful” components from images, also called features, are used. The features are designed to contain colour, edge, corner and texture information. E.g., while RGB-only patches contain colour information, Histogram of Oriented Gradients (HoG) features capture structured edge information and Deep features capture edge and texture information.

RGB features

RGB features are composed of R, G and B channels of the image. These are simplest features having pixel intensity values for each colour channel. Here the features are the images itself.

HoG features

HoG features have shown [44] to be state of the art features for object detection until the advent of data-driven (deep) features. HoG features retain shape information by capturing edge information. Due to localized normalization and binning, it is robust to local variations in illumination and articulation. The distribution of directions of gradients is used as HoG features. Gradients (horizontal and vertical) of an image are useful because the magnitude of gradients is large around edges and corners which are the regions of abrupt intensity changes. Therefore, edges and corners contain valuable information about object shape.

HoG feature extraction is a three-step process: During the first step, first the horizontal, g_x and vertical, g_y gradients are calculated. This is achieved by filtering the image with the Sobel [82] filters. The gradient image has a high response at edges or object boundaries and low response in uniformly coloured regions. Next, The magnitude and direction of the gradient is computed as

$$\begin{aligned} g &= \sqrt{g_x^2 + g_y^2} \\ \theta &= arctan\frac{g_y}{g_x} \end{aligned} \tag{4.1}$$

Where g is the gradient magnitude and θ is the gradient direction. As for colour images, the maximum gradient magnitude among the three channels and the corresponding gradient direction is retained.

During the second step, the image is divided into 8×8 cells and a histogram of gradients is calculated for each 8×8 cell.

In the final step, features are normalised to be independent of local lighting variations. This is illustrated in figure 4.2. The figure shows the nine channelled HoG features with (a) showing all horizontal gradients, (b) incorporating neck highlighting gradients, (c) incorporating face and leg highlighting gradients, (d) incorporating face and leg highlighting gradients, (e) incorporating vertical gradients, (f,g,h) incorporating body and some background highlighting gradients and (i) incorporating background highlighting gradients.

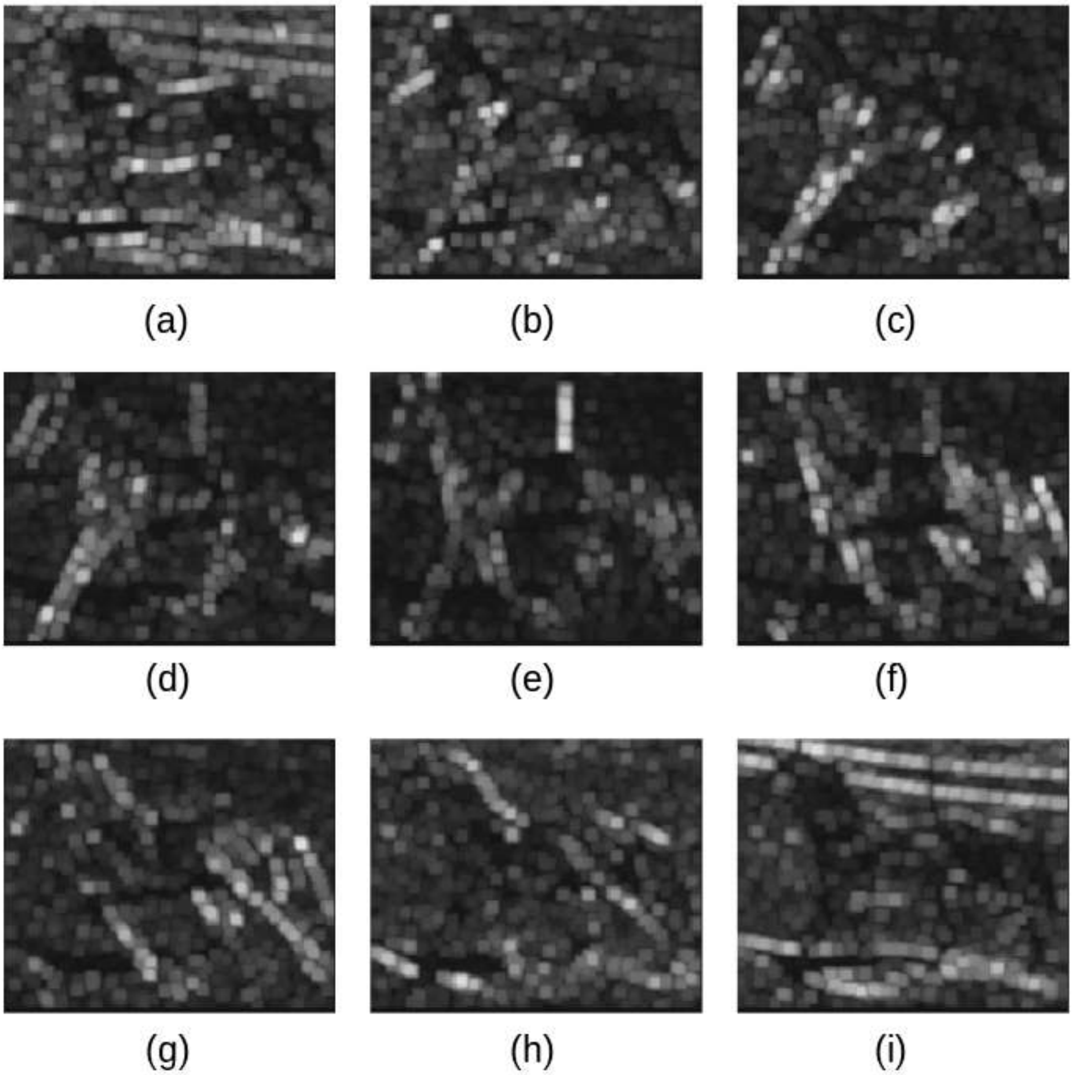


Figure 4.2: The figure shows the nine channelled HoG features of input figure A.1 (a) showing all horizontal gradients, (b) incorporating neck highlighting gradients, (c) incorporating face and leg highlighting gradients, (d) incorporating face and leg highlighting gradients, (e) incorporating vertical gradients, (f,g,h) incorporating body and some background highlighting gradients and (i) incorporating background highlighting gradients.

Deep features

Deep features are extracted using Convolutional Neural Networks (CNNs) which produce hierarchical trainable features. These are robust [9] compared to handcrafted HoG features. The layers of CNN have three dimensions - width, height and depth (activation volume). CNNs make use of the property that if one filter is used to compute some feature, then the same filter should also be useful to compute similar feature at a different position. This property, called parameter sharing, also makes CNNs translation invariant by detecting features regardless of their position in the image. CNNs also take into consideration that images are spatially locally correlated by having a local connectivity pattern between neurons of adjacent layers. The CNN consists of multiple convolutional, rectified linear unit, pooling, dropout and fully connected layers. In this work, features are extracted from the convolutional layer.

The convolutional layer consists of kernels/filters which have a receptive field. During forward pass, the filter convolves around the image computing the dot product between the entries of the filter and the input producing 2-dimensional "activations" of the filter. These filters learn specific features which activate when similar features are found elsewhere in the image. There can be many such filters in a convolutional layer. Stacking these "activations" for all filters along the depth dimension forms the full output of that convolution layer. Visualization of the Convolutional layer's filters can be seen in figure 4.3 and activations in figure 4.4.

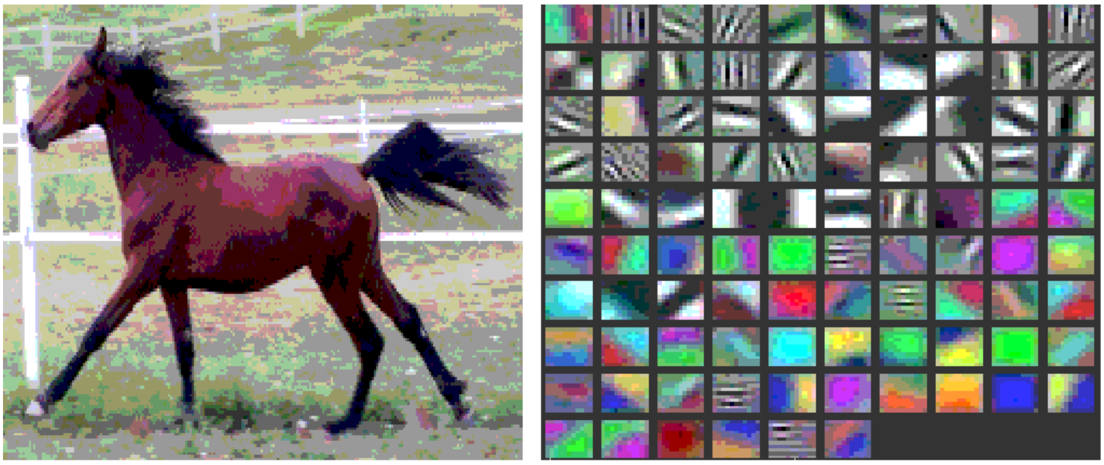


FIGURE 4.3: The input image on left and visualization of the Convolutional layer's filter on right.

The convolutional layer can have hyper-parameters like Receptive field F or size of the filter, the number of filters K and Stride S with which filters slide. When the stride is 1, then filters are moved one pixel at a time, Size of zero-padding P padding the input volume with zeros around the border. It allows us to control the spatial size of the output of the filter.
If the filter receives an input volume of size W1×H1×D1, it produces a volume of size W2×H2×D2 where:

$$\begin{gathered} W2 = \frac{W1 - F + 2P}{S} + 1 \\ H2 = \frac{H1 - F + 2P}{S} + 1 \\ D2 = K \end{gathered} \tag{4.2}$$

With parameter sharing, it introduces $F \times F \times D1$ weights per filter, for a total of $(F \times F \times D1) \times K$ weights and K biases. In this work, VGG19 model trained on

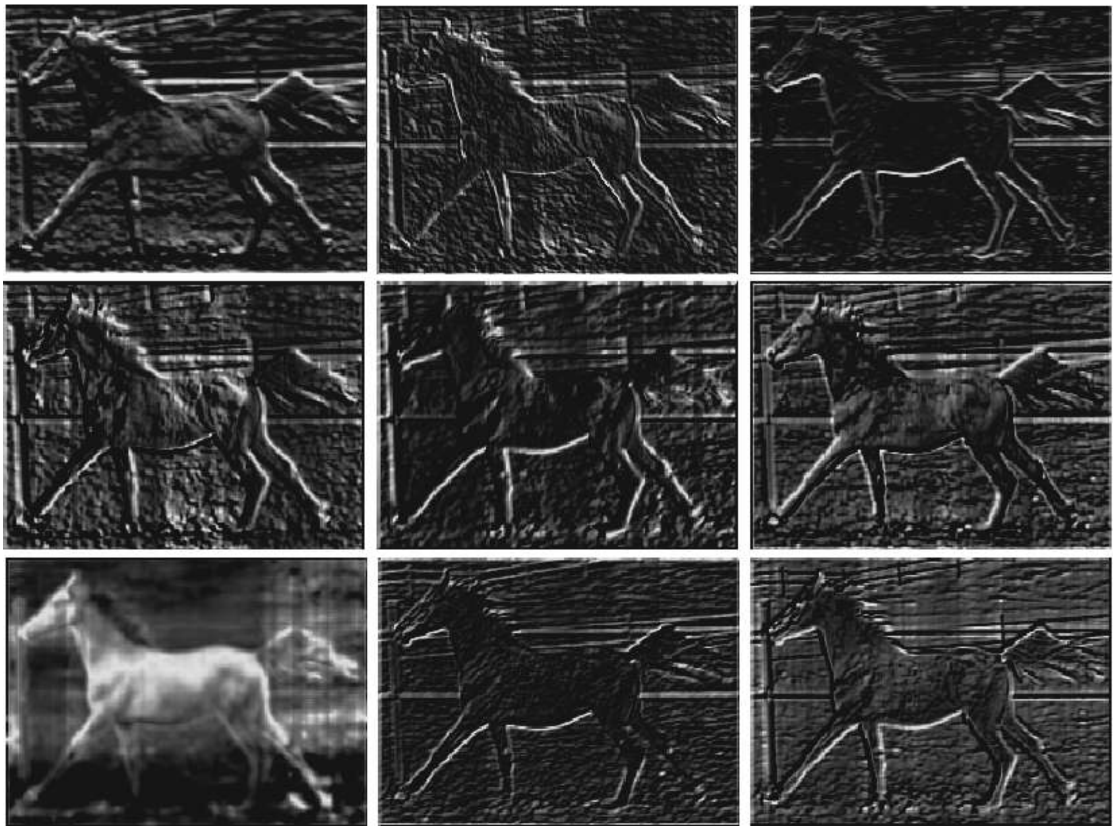


Figure 4.4: Visualization of the Convolutional layer's activations.

Imagenet [9] dataset is used. The features are extracted from the second convolutional layer of this VGG model as shown in figure A.2.

4.1.2 Training

During the training of Hough forests, number of trees and depth of each tree was fixed at 15. The effect of changing these parameters are explained in section 5. The number of tests per each non-leaf node in the tree was fixed to 2000. The non-leaf nodes were randomly chosen to be classification or regression nodes. In each leaf, the class probability and a list of offset vectors are stored. The patch size of the patches extracted from the images from the datasets are fixed to height and width of 16 pixels. The number of patches extracted per image is dependent on the dataset chosen. The stopping criteria for training is when the maximum depth is reached or the number of patches reaching the node is smaller than 2.

4.1.3 Testing

During testing, patches are extracted from the entire test image in multiple scale space. The number of scales is different for different datasets which are dependent on the variety of object shapes present. To find the peaks in Hough space, non-maximal suppression

[83] is performed to find n peaks for n objects. The number of objects n is pre-determined depending on the dataset.

Bounding boxes from Hough space

The bounding box height and width is fixed to be median of height and width of all bounding boxes seen during training. For each scale, the bounding boxes are rescaled accordingly. After non-maximal suppression performed for each peak, a detection is bounding box with centre at the peak.

4.2 Interactive Learning stage

As discussed in section 3, the Active Learning framework is initialized by annotating two images. The first image is used to seed the Hough Forest-based object detector. The scores from the second image are used for seeding the Active Learning phase.

The Active Learning pipeline is implemented as a procedure illustrated in figure 4.5 and steps are as follows:

1. The dataset for Active Learning is chosen.
2. 20% of the dataset is held out for testing and 80% of the dataset is held out for training.
3. Two annotated images or positive images and two negative images (the images which do not contain the object of interest) are selected to train.
4. Interactive detection begins by training on two positive and two negative images.
5. Active learning is performed using bounding box scores obtained by rest of the training images and trained forest model.
6. A beneficial image is chosen from rest of the images whose annotation cost is highest and used for re-training.
7. Evaluation is performed on evaluation images with the trained forest model.
8. Steps 4 to 7 are repeated until the annotation cost is zero.

4.3 Evaluation measure

Object detection

The evaluation metric for object detection is mean Average Precision (mAP). This is calculated as the area under the interpolated precision and recall curve.

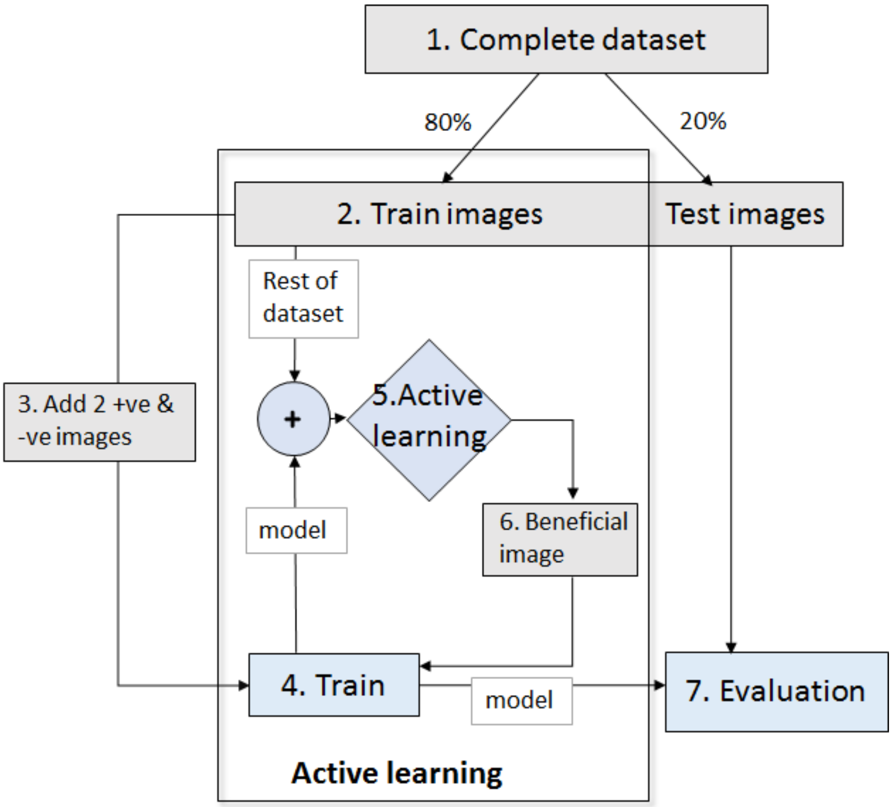


FIGURE 4.5: The Active Learning evaluation pipeline showing steps during Active Learning and evaluation. In step 1, the dataset is obtained. In step 2, the dataset is divided into training and test images. In step 3, two annotated images and two unlabelled images are selected to train. In step 4, 5 and 6, training, active learning and evaluation are performed iteratively until stopping criteria is reached.

		Actual class	
		Positive class	**Negative class**
Predicted class	**Positive class**	True positive	False positive
	Positive class	False negative	True negative

TABLE 4.1: True/False Positive and Negative definition.

To calculate precision and recall, first True Positives, False Positives and False Negatives are evaluated. This is evaluated by using the Intersection over Union (IoU) metric where True Positives have $IoU \geq 0.5$, False Positives have $IoU < 0.5$ and False Negatives are un-detected object instances. The definitions are tabulated in table 4.1. The IoU between a detected bounding box d and a ground-truth bounding box g is calculated as

$$IoU(d, g) = \frac{d \cap g}{d \cup g} \tag{4.3}$$

where $d \cap g$ and $d \cup g$ are areas of intersection and union between bounding boxes d and g respectively. This is graphically illustrated in figure 4.6

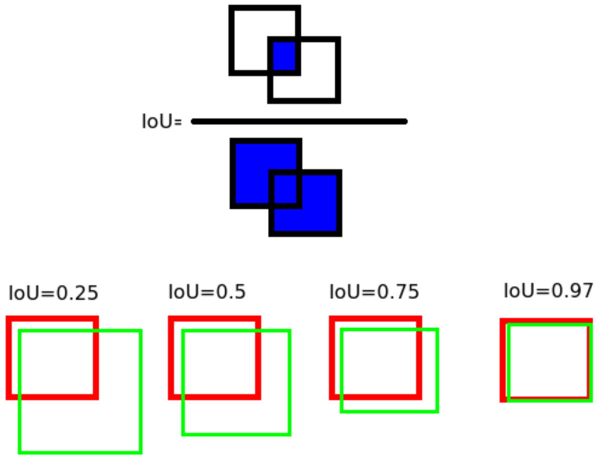


FIGURE 4.6: Figure showing a graphical illustration of Intersection of Union (IoU) and random bounding boxes with IoU of 0.25, 0.5, 0.75 and 0.97.

Precision is calculated as the fraction of positive detections among all detections:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{4.4}$$

and recall as the fraction of positive detections among all possible positive detections:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{4.5}$$

the precision-recall curve shows the tradeoff for varying thresholds of voting confidence. A high value of the area under the curve shows both high recall and high precision. A high precision value corresponds to low false positive detections and high recall value corresponds to low false negative detections. If a model has a high recall and low precision it results in many detections but most of its detections are incorrect. If a model has high precision and low recall it results in few but correct detections. Ideal object detection model has high precision and recall which results in many and correct detections.

The mAP is then calculated as the area under the interpolated precision-recall curve, as illustrated in figure 4.7 (hashed region).

Active learning

The effectiveness of Active Learning is found by comparing Active Learning to incremental/passive learning where new samples to be annotated are selected randomly. Further, Active Learning experiments are performed until stopping criteria, i.e. when the annotation cost is zero, is reached.

Hard Negative mining

A popular trick of boosting object detection performance is to incorporate hard negative mining [84]. Here, false positives from the previous iteration are added as negative examples during training the model in the present iteration. This helps the Hough

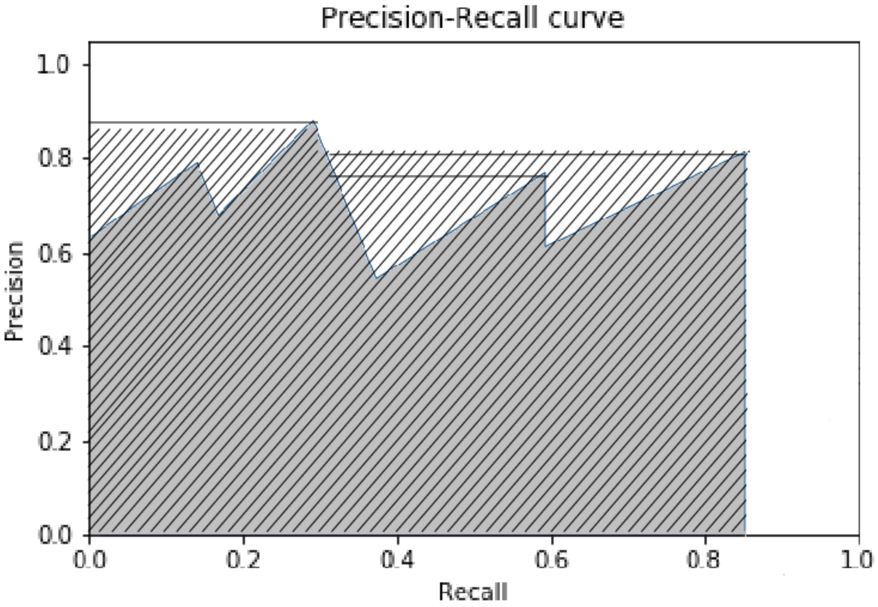


FIGURE 4.7: Interpolated precision-recall curve for example iteration.

Forest to distinguish between real positive samples and difficult negative samples. This technique is incorporated in all experiments discussed in section 5.

4.4 Adaption in web annotation tool

To obtain annotations from the user in every iteration, a web-based annotation tool is developed. It is adapted from [85] using Python [86] programming language and Flask framework [87]. The snapshot of the tool is shown in figure 4.8. It has the following functionalities

1. "Start" experiment by loading two images.
2. Annotate object of interests with ground truth bounding boxes.
3. Download the annotations and "Train" the Hough Forests.
4. Perform Active Learning using "Test" button.
5. Load the beneficial image.
6. Repeat steps 3-5 n times to annotate n beneficial images.

To the available annotation application, the functionality of Active Learning has been added. In figure 4.8, the top black task bar has "Start" button to load first two images.

The image annotation is done using the left side tool bar by choosing the annotation bounding box of our interest. The shape of the bounding box used is the rectangle. The annotated images are trained using the "Train" button on the top black task bar. For the completely annotated datasets, a graph with mAP for the trained images is shown below the black task bar (not shown in the figure). The graph also shows the comparison of mAP of Fully Supervised, Passive and Active Learning. The graph is updated every time after training. Active learning is performed by clicking on "Test" button on the top black task bar. "Load" button is used to load the beneficial image which can be used in next training and Active Learning iteration. The code for this tool is available in the CD attached.

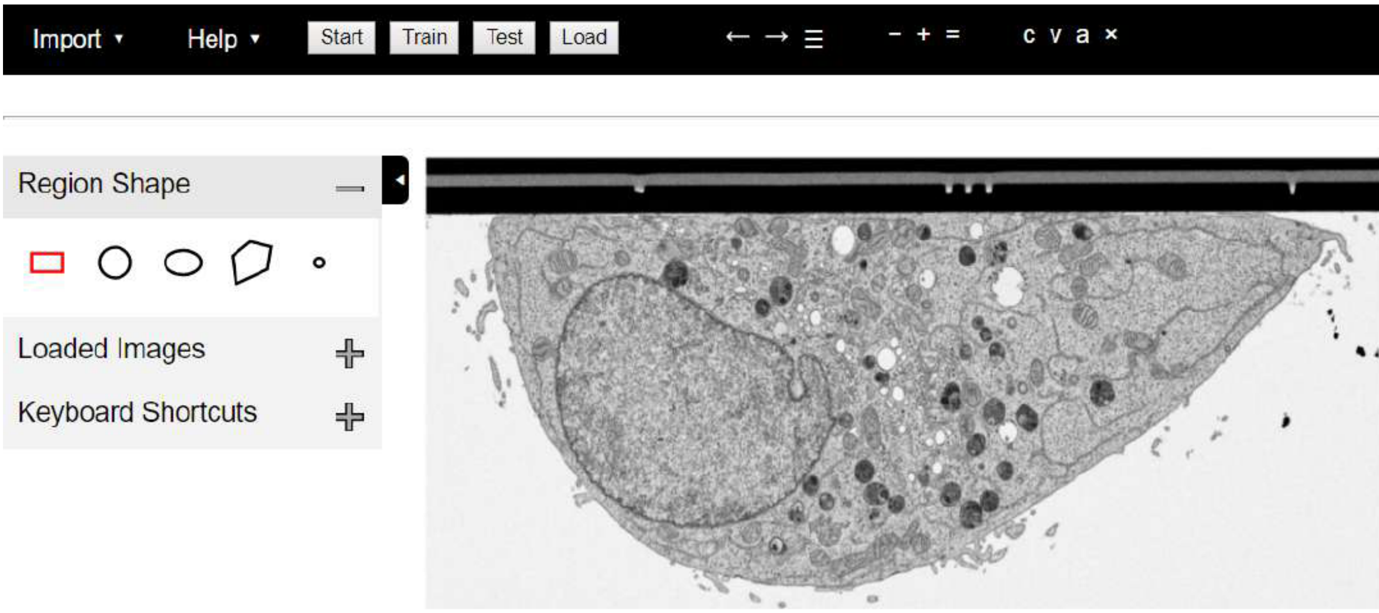


FIGURE 4.8: Snapshot of web annotation tool. The top black task bar has ”Start” button to load first two imag using the left side tool bar by choosing the annotation bounding box of our interest. The annotated images are on the top black task bar. Active learning is performed by clicking on ”Test” button on the top black task bar. ” beneficial image which can be used in next training and Active Learning iteratio

Chapter 5

Evaluation

In this chapter, details of the datasets, experimental analysis, qualitative and quantitative results are presented.

5.1 Experimental setup

5.1.1 Datasets

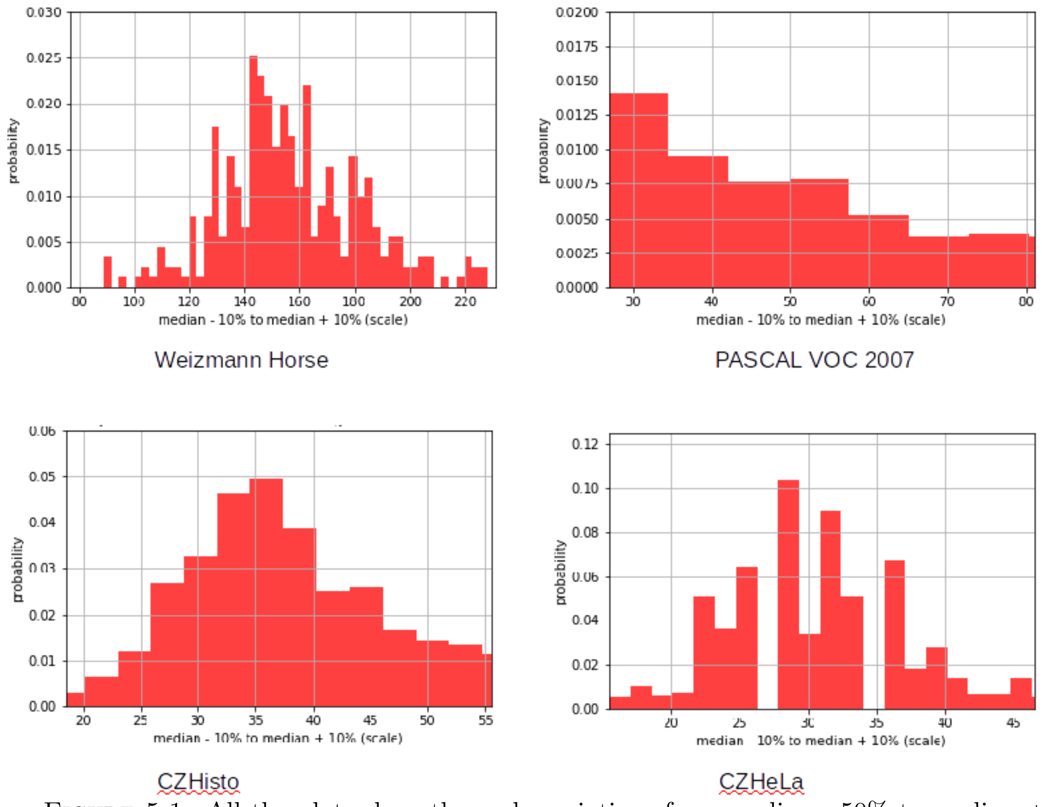
Four datasets are considered for experimental evaluation of Active Learning and object detection. Two of these datasets are standard benchmark datasets used in the computer vision community. The Weizmann Horse [88] is a relatively small RGB-dataset containing horses as the object of interest. The PASCAL VOC 2007 [89] dataset is closer to a medium-sized real-world RGB-dataset. The class "car" is considered for experimental evaluation as shown in [30]. Both datasets come bundled with ground truth bounding box annotations. Sample images from both datasets are illustrated in figure 5.4 and figure 5.8. The next two datasets are custom Carl Zeiss datasets. CZHisto is a Histopathology RGB-dataset where objects of interest are steatosis cells. Finally, CZHeLa [90] is a dataset captured by a Focussed Ion Beam -Scanning Electron Microscope (FIB-SEM) of cervical cancer HeLa (Henrietta Lacks) cell where objects of interest are mitochondria. Sample images from this dataset are illustrated in figure 5.14. Each image is a single-channel image stack. Ground truth annotations for these datasets were obtained through manual annotation. The properties of various datasets are summarized in table 5.1.

The scale variations for the four datasets are shown in figure 5.1. For each dataset, the median of minimum value among height and width of the bounding boxes is found. In the figure 5.1, all the plots show median - 50% to median + 50% of scale variations. The plots which have Gaussian nature represent lesser scale variations and others represent higher scale variations. Clockwise, the first plot shows the scale variations for Weizmann Horse

	Name	Size	Object	# Objects	Colour	Challenges
Computer vision benchmark	**Weizmann Horse**	328	horse	1	yes	easy
	PASCAL VOC 2007	761	car	1-5	yes	occlusions, scale variations
Carl Zeiss datasets	**CZHisto**	600	steatosis cells	∼ 15	yes	unclear object boundaries
	CZHeLa	120	mitochondria	∼ 20	grayscale	scale variations

Table 5.1: Characteristics of datasets used in the experiments.

dataset. Second shows the scale variations for PASCAL VOC 2007 dataset. Third shows scale variations for CZHeLa dataset and fourth shows variations for CZHisto dataset. Among all the plots, PASCAL VOC 2007 shows highest scale variation and is also the most difficult dataset.



Weizmann Horse

PASCAL VOC 2007

CZHisto

CZHeLa

Figure 5.1: All the plots show the scale variations from median - 50% to median + 50% of scale variations. The plots which have Gaussian nature represent lesser scale variations and others represent higher scale variations. Clockwise, the first plot shows the scale variations for Weizmann Horse dataset. Second shows the scale variations for PASCAL VOC 2007 dataset. Third shows scale variations for CZHeLa dataset and fourth shows variations for CZHisto dataset. Among all the plots, PASCAL VOC 2007 shows highest scale variation.

5.1.2 Experimental baselines

In order to know the impact of using RGB, HoG and Deep features, Fully Supervised Learning is conducted on each dataset. Fully Supervised learning uses all annotated training images. The best feature is chosen to conduct experiments using Active Learning. Further, Active Learning is compared to Incremental/Passive Learning where new images to be annotated are obtained through random sampling. However, in Active Learning, the beneficial images are picked according to equation 3.15 in every iteration. The iterative procedure of Active Learning terminates when the annotation cost is zero.

5.1.3 Image augmentation

To check the effects of expanding the number of images in the dataset, image augmentation is performed. Image augmentation is a process of expanding dataset by slightly altering the existing images. The alteration can be done by flipping, inverting, blurring, adding Gaussian noise and affine transformations. In this work, images are added by flipping the images left to right and up to down. A Fully Supervised experiment is performed on 1000 images extended from Weizmann Horse dataset. Some images from the extended dataset are shown in figure 5.2. mAP of original dataset and with data augmentation are compared in table 5.2. From the table, it can be seen that image augmentation does not contribute to increased performance. So image augmentation is not conducted on all datasets during experiments.

	Original dataset	**With augmentation**
Avg. mAP	0.95	0.93

Table 5.2: Results of data augmentation on Weizmann Horse dataset.

5.1.4 Cross-Validation

When multiple experiments are performed on same train and test set of the dataset, there are chances that the model learns the dataset instead of generalising to an independent dataset. This is called overfitting. In cross-validation, the dataset is partitioned into different train and test sets in multiple rounds and in each round the experiments are performed. The results are averaged over rounds to estimate the performance of the model. In the experiments, cross-validation is performed for Fully Supervised Learning.

5.1.5 Tools

Python programming language version 3.0 is used to program Active Learning with object detection. Fertilised Forests python library's Hough Forest code [91] based on

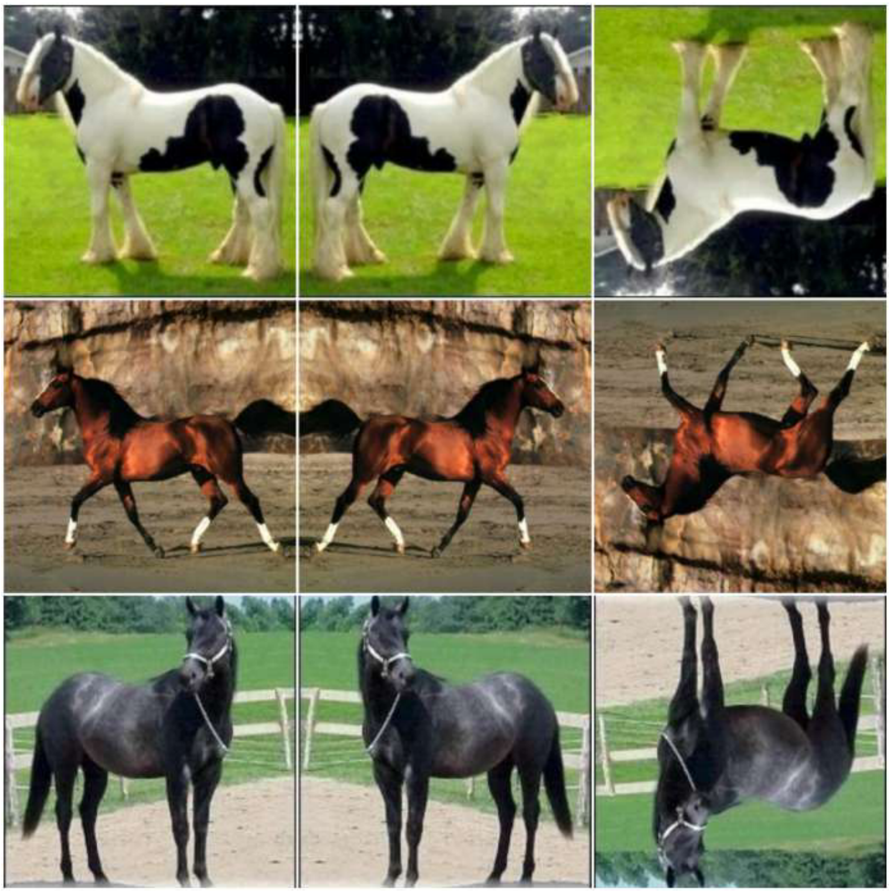


FIGURE 5.2: Example images from data augmentation performed on Weizmann Horse dataset.

work [30] is used and the Active Learning is implemented according to the method in [1]. Annotation tool is adapted from web annotation tool [85] according to our needs using Python programming language and Flask framework [92].

The hardware used is 64 bit, 8 GB RAM machine with core i5 2.50GHz processor and GTX1080 GPU support only for Deep Learning experiments.

5.1.6 Hough Forest hyperparameters

The hyperparameters in Hough Forests are number of trees, depth of each tree, number of patches extracted from images and number of scales in Hough space. The most efficient values of hyperparameter are chosen to conduct final experiments. The effect of varying hyperparameters on Weizmann Horse dataset is explained.

- **Depth of trees in forest**. From the experiments, the performance is best at tree depth 10. For the experiments, tree depth of 15 is chosen. A comparison with increasing tree depth is shown in figure 5.3 (a).

- **Number of trees in forest**: From the experiments, at least 5 trees are needed to achieve good performance. After having 15 trees, the performance does not change much. The number of trees is fixed 15 in the experiments. A comparison with increasing number of trees is shown in figure 5.3 (b).

- **Number patches extracted**: A Higher number of patches encode more information from the images. Best performance is achieved from 20 patches per image and the performance saturates later. A comparison with increasing number of patches is shown in figure 5.3 (c).

- **Number of scales in Hough space**: As the number of scales increase, the objects of varying size can be detected. This hyperparameter is dataset dependant and for Weizmann horse dataset, this number is found to be 3. The performance falls if the number of scales is increased beyond 3. A comparison with increasing number of scales is shown in figure 5.3 (d).

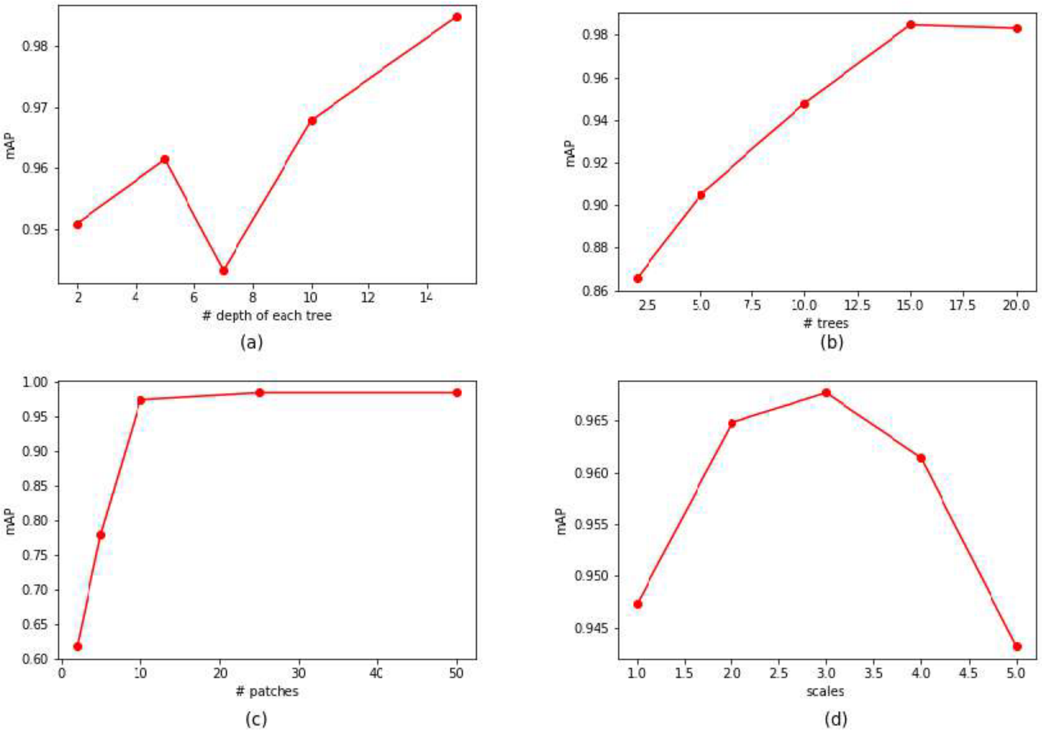


FIGURE 5.3: Effect of various hyperparameters on Active Learning. (a) Depth of trees in forest. (b) Number of trees in forest. (c) Number patches extracted (d) Number of scales in Hough space.

5.1.7 Time taken in experiments

The time taken to train each tree is found to be 0.40 seconds. To compute optimal threshold, prediction and annotation cost in every iteration, for every image it takes 0.35 seconds.

	Fully Supervised mAP		
Cross-validation set	**RGB**	**HoG**	**Deep features**
1	0.90	0.96	0.98
2	0.91	0.95	0.99
3	0.91	0.95	0.98
4	0.91	0.95	0.98
5	0.91	0.95	0.99
Avg.	0.91	0.95	**0.98**

Table 5.3: 5-way cross-validation results obtained by Fully Supervised Learning using RGB, Hog and Deep features for Weizmann Horse dataset.

5.2 Results

The qualitative and quantitative results of all four datasets are present here.

5.2.1 Computer Vision benchmark datasets

Benchmark datasets have been released to stimulate the research in the field of computer vision. Various computer vision algorithms perform experiments on this benchmark datasets to compare their performance.

5.2.1.1 Weizmann Horse

Nature of dataset
This dataset has 328 images of horses which are coloured, in side-view. One image has only one instance of horse object. The images are mostly collected from the Internet and Caltech [93] database. The images from the Internet are cropped by hand. Figure 5.4 shows example images from the dataset. 262 images are used for training and 66 images for testing. Experiments show 25 patches per image are sufficient to obtain state-of-the-art results.

Fully Supervised learning
All the 262 images are used for Fully Supervised training with Hough forests. The trained forest model is used to evaluate on 66 images. Fully Supervised learning is performed with RGB, HoG and Deep features extracted from images. Table 5.3 shows results from the fully-supervised learning experiments with 5-way cross-validation. Figure 5.5 is a bar graph showing the mean mAP of RGB, HoG and Deep features. This figure shows that Deep features perform the best among other features with mAP 0.98. These features are used for conducting Active Learning experiments.

Active learning
Active learning is conducted with the same validation set as Fully Supervised Learning

Figure 5.4: Example images from Weizmann Horse dataset.

for a fair comparison of scores. 10 experiments are performed with randomly chosen first two images to train. In each experiment, Active Learning and Passive Learning experiments are conducted. Figure 5.6 shows the results of experiments. After 5 iterations, i.e after annotating 7 images, the annotation cost becomes zero. It can be observed that Active and Passive Learning perform similarly due to the simple nature of the dataset where there is not much intra-dataset variation. However, Active Learning slightly performs better than Passive Learning. It is observed that Active Learning uses only 2% of the annotated dataset to achieve mAP comparable to Fully Supervised Learning.

Model Training Times

In table 5.4 a comparison of time taken to perform Fully Supervised Learning and Active Learning is made. For Active Learning experiment, a total of 7 images are annotated and time taken to annotate each image is 2.5 seconds. So, in total, the time taken to annotate is 7 × 2.5 which is equal to 17.5 seconds. Time taken to train is 15 × 0.4 which is equal to 6 seconds and time taken to find the beneficial image is 262 test images × 0.35 is equal to 91.7 seconds. In total, the time taken for the Active Learning experiment is 17.50 + 6.0 + 91.7 = 115.2 seconds. For Fully Supervised experiments, the time taken to annotate 328 images is equal to 820 seconds and time taken for the experiment is 66 test images × 0.35 which is equal to 23.10 seconds. So total time taken for Fully Supervised

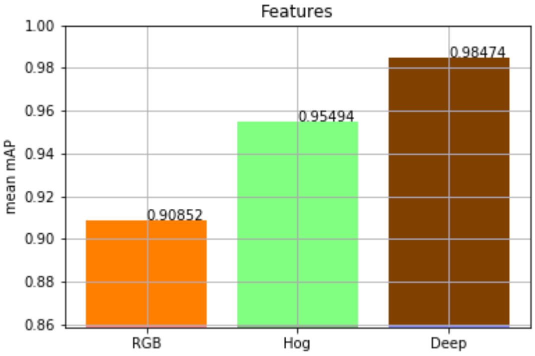


FIGURE 5.5: Fully Supervised mAP for RGB, HoG and Deep features using Weizmann Horse dataset.

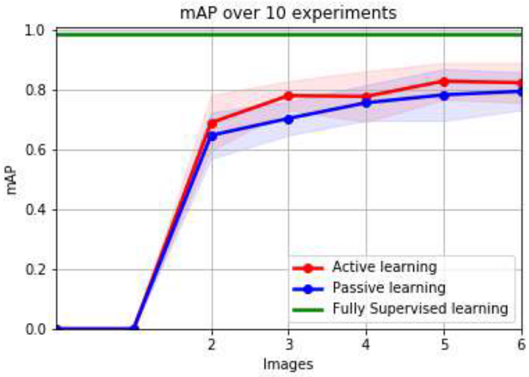


FIGURE 5.6: Fully Supervised vs Passive vs Active learning, Deep features using Weizmann Horse dataset.

	Time taken (s)
Fully Supervised	849.10
Active learning	115.2

TABLE 5.4: Comparison of time taken to conduct Fully Supervised Learning vs Active Learning experiments for Weizmann Horse dataset.

	Avg. precision
Hough Forests + HoG [30]	0.96
This work	0.98

TABLE 5.5: State-of-the-art object detection comparison for Weizmann Horse dataset.

experiments and its annotation is 820 + 23.10 + 6.0 is equal to 849.10 seconds. The table shows Active Learning experiment results in 7.4x speed up.

State-of-the-art comparison

A previous work [30] using Hough Forests with HoG features reports a mAP 0.96 for same hyperparameter settings (with varying aspect ratio values, a mAP of 0.98 is achieved). This work with the same hyperparameter settings, but with Deep features achieves average mAP of 0.98474. Therefore it is verified that Deep features can improve object detection performance by 2%. A comparison canbe seen in table 5.5.

Qualitative results

Figure 5.7 shows qualitative results from Weizmann Horse dataset. Cyan boxes show False Negative detections, blue boxes show False Positive detections and red boxes show True Positive detections. The first image in each row has no positive detections. By inspection, it is verified that if the horse does not cover the entire image or horse is not completely in side-view, the detections are sometimes not positive.

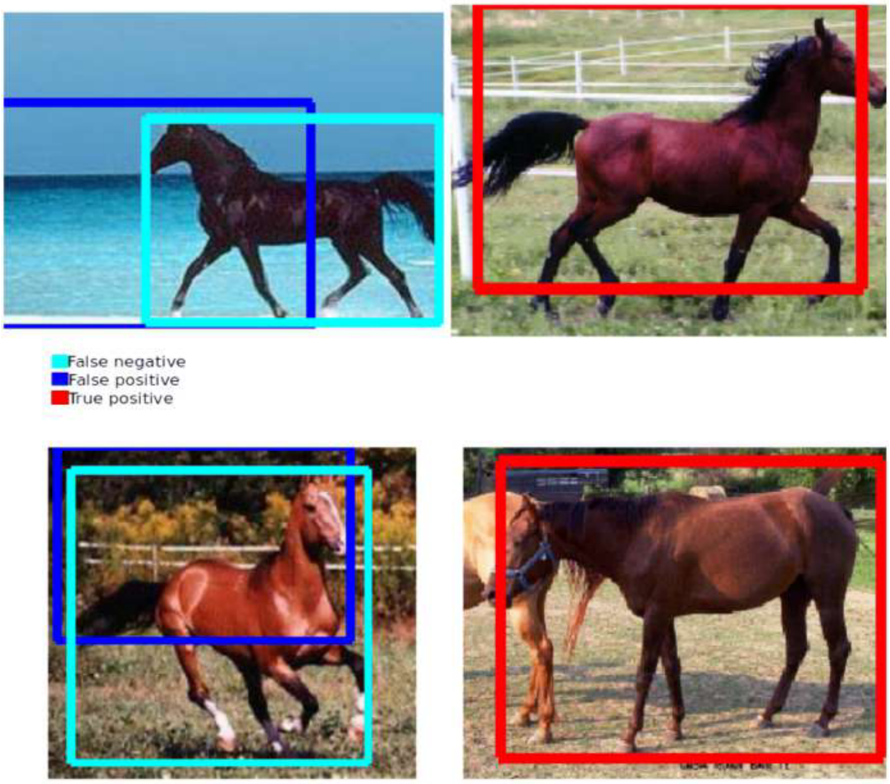


FIGURE 5.7: Qualitative results from Weizmann Horse dataset.

5.2.1.2 PASCAL VOC 2007

Nature of dataset

The original dataset has 21 objects but only car object is chosen from the dataset. This dataset has 761 images of cars which are of different colours and shapes. One image has multiple instances of car object. The PASCAL VOC project provides standardised image data sets for object class recognition. The main goal of PASCAL VOC 2007 challenge is to recognize objects from a number of visual object classes in realistic scenes (i.e. not pre-segmented objects). This is a difficult dataset with multiple objects in one image of different shapes and with occlusions. Figure 5.8 shows example images from the dataset. 609 images are used for training and 152 images for testing. Experiments show 200 patches are sufficient to obtain state-of-the-art results.

Quantitative results

Fully Supervised learning

All the 761 images are used for Fully Supervised training with Hough forests. The trained forest model is used to evaluate on 152 images. Fully Supervised learning is performed with RGB, HoG and Deep features extracted from images. Table 5.6 shows results from the fully-supervised learning experiments with 5-way cross-validation. Figure 5.9

Figure 5.8: Example images from PASCAL VOC 2007 dataset.

	Fully Supervised mAP		
Cross-validation set	**RGB**	**HoG**	**Deep features**
1	0.16	0.08	0.15
2	0.15	0.10	0.16
3	0.13	0.18	0.16
4	0.11	0.17	0.16
5	0.13	0.09	0.18
Avg.	0.14	0.13	**0.16**

Table 5.6: 5-way cross-validation results obtained by Fully Supervised Learning using RGB, HoG and Deep features for PASCAL VOC dataset.

is the bar graph showing the mean mAP of RGB, HoG and Deep features. This figure shows that Deep features perform the best among other features with mAP 0.16. These features are used for conducting Active Learning experiments.

Active learning

Active learning is conducted with the same validation set as Fully Supervised Learning for a fair comparison of scores. 10 experiments are performed with randomly chosen first two images to train. In each experiment, Active Learning and Passive Learning experiments are conducted. The figure 5.10 shows the results of experiments. After

	Time taken (s)
Fully Supervised	3864.2
Active learning	254.5

Table 5.7: Comparison of time taken to conduct Fully Supervised Learning vs Active Learning experiments for PASCAL VOC 2007 dataset.

5 iterations, i.e. after annotating 7 images, the annotation cost becomes zero. Active learning outperforms Passive Learning. It is observed that Active Learning uses only 1% of the annotated dataset to achieve mAP comparable to Fully Supervised Learning.

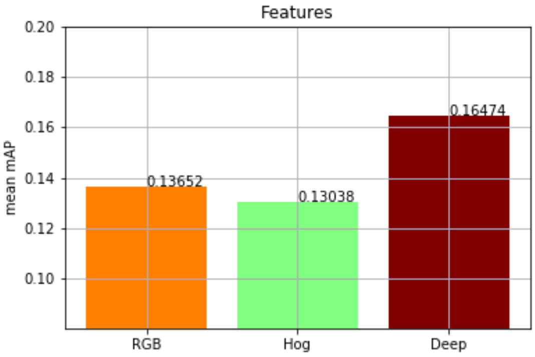


Figure 5.9: Fully Supervised mAP for RGB, HoG and Deep features using PASCAL VOC 2007 dataset.

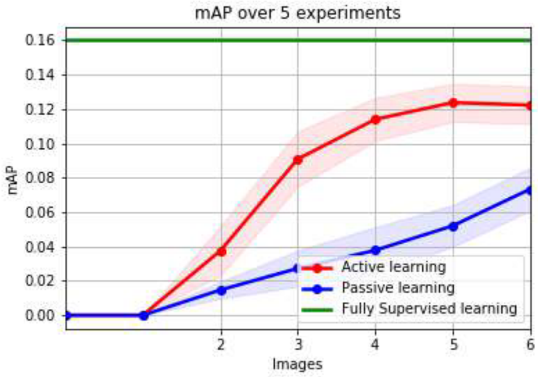


Figure 5.10: Fully Supervised vs Passive vs Active learning, Deep features PASCAL VOC 2007 dataset.

Time taken

In table 5.7 a comparison of time taken to perform Fully Supervised Learning and Active Learning is made. For Active Learning experiment, a total of 7 images are annotated and time taken to annotate each image is 5 seconds. So, in total, the time taken to annotate is 7 × 5 is equal to 35 seconds. Time taken to train is 15 × 0.4 which is equal to 6 seconds and time taken to find the beneficial image is 609 test images × 0.35 is equal to 213.15 seconds. In total, the time taken for the Active Learning experiment is 35 + 6.0 + 213.15 = 254.5 seconds. For Fully Supervised experiments, the time taken to annotate 761 images is equal to 3805 seconds and time taken for the experiment is 152 test images × 0.35 which is equal to 53.2 seconds. So total time taken for Fully Supervised experiments and its annotation is 3805 + 53.2 + 6.0 is equal to 3864.2 seconds. The table shows Active Learning experiment results in 15.18x speed up.

State-of-the-art comparison

Deep learning state-of-the-art methods report mAP of 0.88. A similar work [30] reports mAP of 0.166 with different aspect ratio values. This work reports a similar mAP of 0.1647. A comparison canbe seen in table 5.8.

Qualitative results

The figure 5.11 shows qualitative results from PASCAL VOC 2007 dataset. Cyan boxes

	Avg. precision
Faster R-CNN [53]	0.88
SSD [13]	0.89
Hough Forests [30]	0.16
This work	0.16

Table 5.8: State-of-the-art object detection comparison for PASCAL VOC 2007 dataset.

show False Negative detections, blue boxes show False Positive detections and red boxes show True Positive detections. In this dataset, there are images with highly varying number of objects, their scales and occlusions. Therefore it results in a lot of False Positive detections (shown in blue).

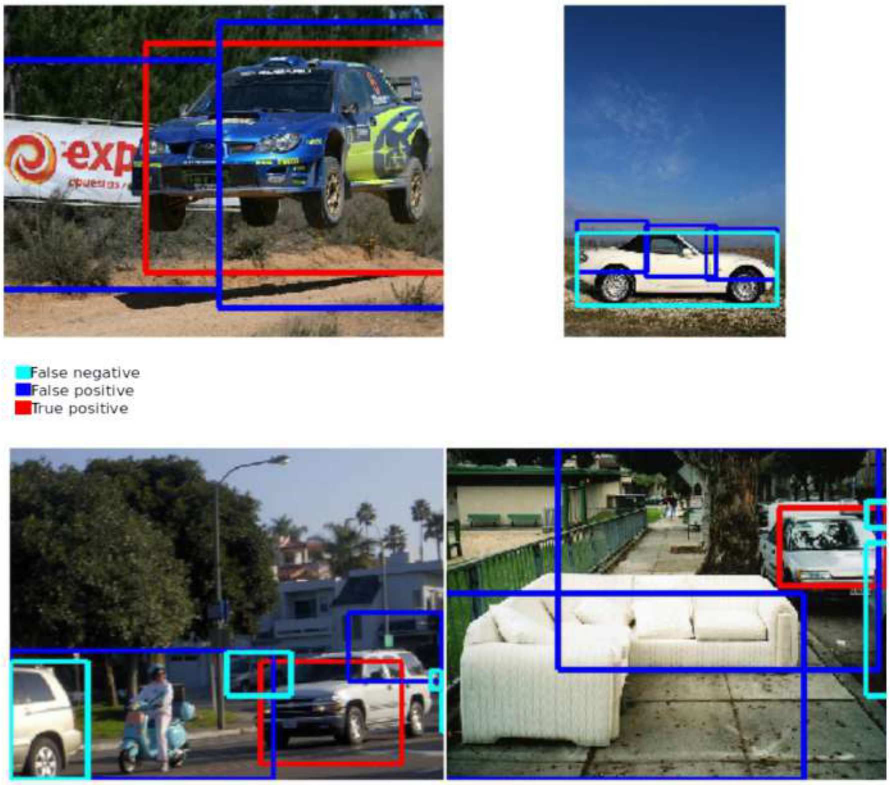


Figure 5.11: Qualitative results from PASCAL VOC dataset.

5.2.2 Carl Zeiss custom datasets

Next, the results from two Carl Zeiss custom datasets are explained.

	Fully Supervised mAP		
Cross-validation set	**RGB**	**HoG**	**Deep features**
1	0.63	0.79	0.81
2	0.70	0.79	0.83
3	0.63	0.80	0.80
4	0.66	0.80	0.82
5	0.60	0.76	0.84
Avg.	0.64	0.79	**0.82**

Table 5.9: 5-way cross-validation results obtained by Fully Supervised Learning using RGB, Hog and Deep features for CZHisto dataset.

5.2.2.1 CZHisto

Nature of dataset

This dataset has 600 images of steatosis (lipid) cells which are white in colour with round shape. One image has multiple instances of steatosis cells. The images are collected at Carl Zeiss. This dataset is hand annotated. This is a medium-level-difficulty dataset with multiple cells in one image and with multiple shapes and unclear object boundaries. 480 images are used for training and 120 images for testing. The images from the dataset cannot be shown due to privacy issues.

Quantitative results

Fully Supervised learning

All the 600 images are used for Fully Supervised training with Hough forests. The trained forest model is used to evaluate on 120 images. Fully Supervised learning is performed with RGB, HoG and Deep features extracted from images. Table 5.9 shows results from the fully-supervised learning experiments with 5-way cross-validation. Figure 5.12 is a bar graph showing the mean mAP of RGB, HoG and Deep features. This figure shows that Deep features perform the best among other features with mAP 0.82. These features are used for conducting Active Learning experiments.

Active learning

Active learning is conducted with the same validation set as Fully Supervised Learning for a fair comparison of scores. 10 experiments are performed with randomly chosen first two images to train. In each experiment, Active Learning and Passive Learning experiments are conducted. The figure 5.13 shows the results of experiments. After 10 iterations, i.e after annotating 12 images, the annotation cost becomes zero. Active learning outperforms Passive Learning. It is observed that Active Learning uses only 2% of the annotated dataset to achieve mAP comparable to Fully Supervised Learning.

Time taken

In table 5.10 a comparison of time taken to perform Fully Supervised Learning and

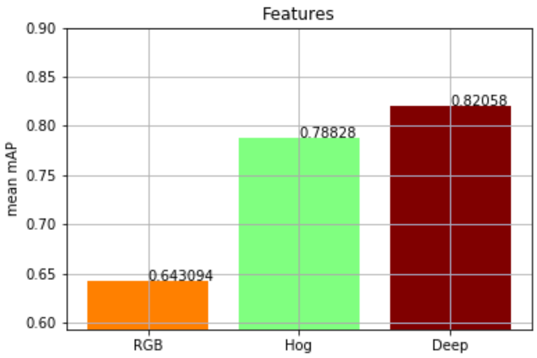


FIGURE 5.12: Fully Supervised mAP for RGB, HoG and Deep features using CZHisto dataset.

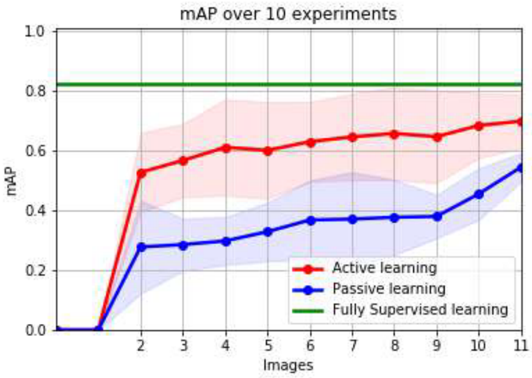


FIGURE 5.13: Fully Supervised vs Passive vs Active learning, Deep features using CZHisto dataset.

	Time taken (s)
Fully Supervised	30046
Active learning	774

TABLE 5.10: Comparison of time taken to hand-annotate dataset for Fully Supervised Learning vs Active Learning, CZHisto dataset.

Active Learning is made. For Active Learning experiment, a total of 12 images are annotated and time taken to annotate each image is 50 seconds. So, in total, the time taken to annotate is 12 × 50 which is equal to 600 seconds. Time taken to train is 15 × 0.4 which is equal to 6 seconds and time taken to find the beneficial image is 480 test images × 0.35 is equal to 168 seconds. In total, the time taken for the Active Learning experiment is 600 + 6.0 + 168 = 774 seconds. For Fully Supervised experiments, the time taken to annotate 600 images is equal to 30000 seconds and time taken for the experiment is 120 test images × 0.35 which is equal to 40 seconds. So total time taken for Fully Supervised experiments and its annotation is 30000 + 40 + 6.0 is equal to 30046 seconds. The table shows Active Learning experiment results in 38.82x speed up.

State-of-the-art comparison

No object detection performance numbers are reported on this dataset.

Qualitative results

The images from the dataset cannot be shown due to privacy issues.

5.2.2.2 CZHeLa

Nature of dataset

This dataset has 120 images of HeLa (cervical cancer) cells which are grayscale images. One image has multiple instances of mitochondria object. The images are collected from FIB-SEM. The images are the slices of one 3-dimensional image of a HeLa cell. The images are hand annotated. This is a difficult dataset of objects of different shapes.

Figure 5.14 shows example images from the dataset. 96 images are used for training and 24 images for testing.

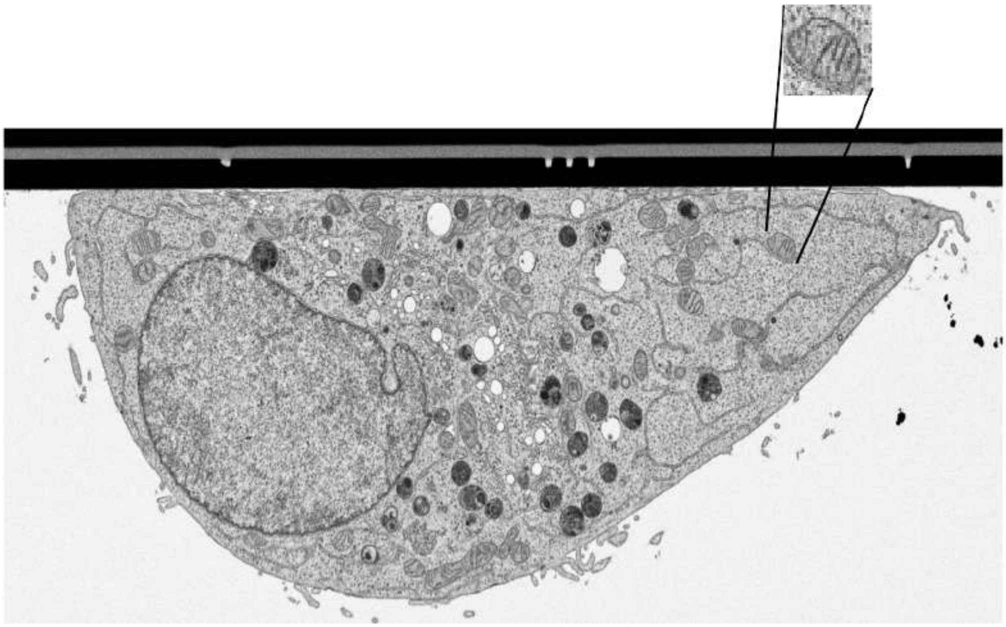


FIGURE 5.14: Example images from CZHeLa dataset.

Quantitative results

Fully Supervised learning

All the 120 images are used for Fully Supervised training with Hough forests. The trained forest model is used to evaluate on 24 images. Fully Supervised learning is performed with RGB, HoG and Deep features extracted from images. Table 5.11 shows results from the fully-supervised learning experiments with 5-way cross-validation. Figure 5.15 is a bar graph showing the mean mAP of RGB, HoG and Deep features. This figure shows that HoG features perform the best among other features with mAP 0.38. The CNN network used is VGG19 which is pre-trained on a coloured dataset, Imagenet. As CZHeLa is a single channelled grayscale image dataset, the deep features do not perform better than HoG features. HoG features are used for conducting Active Learning experiments.

Active learning

Active learning is conducted with the same validation set as Fully Supervised Learning for a fair comparison of scores. 10 experiments are performed with randomly chosen first two images to train. In each experiment, Active Learning and Passive Learning experiments are conducted. The figure 5.16 shows the results of experiments. After 10 iterations, i.e. after annotating 12 images, the annotation cost becomes zero. Active learning outperforms Passive Learning. It is observed that Active Learning uses only 10% of the annotated dataset to achieve mAP comparable to Fully Supervised Learning. However, it can be seen that Active Learning performance is lower than Fully Supervised

	Fully Supervised mAP		
Cross-validation set	**RGB**	**HoG**	**Deep features**
1	0.10	0.40	0.31
2	0.10	0.41	0.36
3	0.10	0.36	0.32
4	0.11	0.37	0.33
5	0.13	0.38	0.32
Avg.	0.11	**0.38**	0.32

TABLE 5.11: 5-way cross-validation results obtained by Fully Supervised Learning using RGB, Hog and Deep features for CZHisto dataset.

Learning. This is due to objects of two different object shapes (two non-overlapping groups of object scores) found when annotation cost is zero.

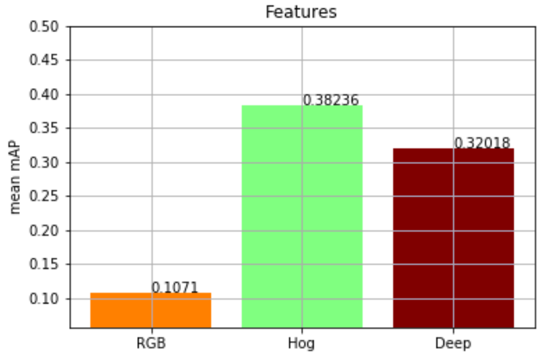


FIGURE 5.15: Fully Supervised mAP for RGB, HoG and Deep features using CZHeLa dataset.

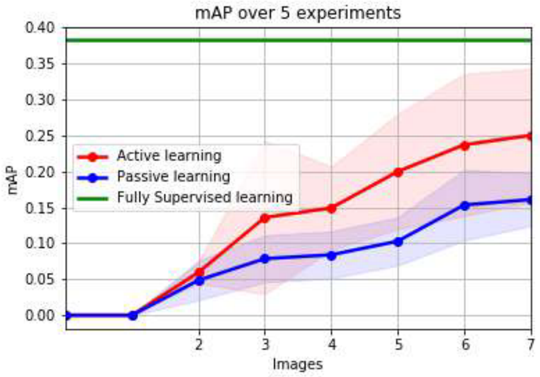


FIGURE 5.16: Fully Supervised vs Passive vs Active learning, HoG features using CZHeLa dataset.

Time taken

In table 5.12 a comparison of time taken to perform Fully Supervised Learning and Active Learning is made. For Active Learning experiment, a total of 12 images are annotated and time taken to annotate each image is 60 seconds. So, in total, the time taken to annotate is 12 × 60 which is equal to 720 seconds. Time taken to train is 15 × 0.4 which is equal to 6 seconds and time taken to find the beneficial image is 96 test images × 0.35 is equal to 33.6 seconds. In total, the time taken for the Active Learning experiment is 33.6 + 6.0 + 720 = 759.6 seconds. For Fully Supervised experiments, the time taken to annotate 120 images is equal to 7200 seconds and time taken for the experiment is 96 test images × 0.35 which is equal to 33.6 seconds. So total time taken for Fully Supervised experiments and its annotation is 7200 + 33.6 + 6.0 is equal to 7239.6 seconds. The table shows Active Learning experiment results in 9.5x speed up.

State-of-the-art comparison

No object detection performance numbers are reported on this dataset.

	Time taken (s)
Fully Supervised	7239.6
Active learning	759.6

Table 5.12: Comparison of time taken to conduct Fully Supervised Learning vs Active Learning experiments for CZHeLa dataset.

Qualitative results

The figure 5.17 shows qualitative results from CZHeLa dataset experiments. Cyan boxes show False Negative detections, blue boxes show False Positive detections and red boxes show True Positive detections. The number of objects per image and their scale vary in the dataset which results in a lot of False Positive detections (in blue).

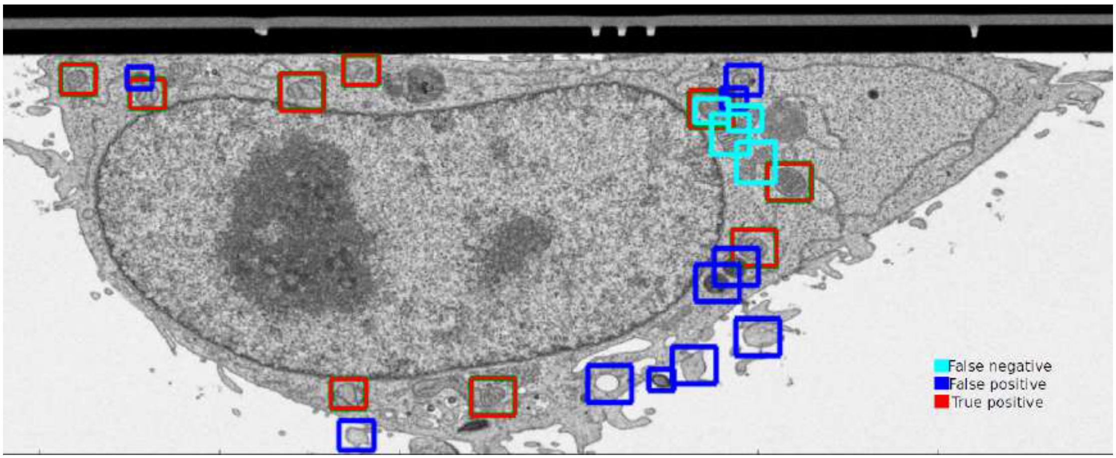


Figure 5.17: Qualitative results from CZHeLa dataset.

Chapter 6

Conclusions and Future work

In this work, an Interactive Object Detection pipeline is realized in order to minimise the trade-off between the effort to annotate (annotation cost) unlabelled data and the performance of object detection model. The detection pipeline is composed of an object detection stage followed by an Active Learning stage performed iteratively. Hough Forests are used for object detection. Given the increased ubiquity of CNNs, the utility of deep features in improving performance is also explored. In Active Learning, to choose the beneficial images, annotation cost is calculated as the predicted false positive and false negative rate. The framework is successfully evaluated on two Computer Vision benchmark datasets i.e. Weizmann Horse and PASCAL VOC 2007 and two Carl Zeiss custom datasets i.e. CZHisto and CZHeLa.

As for image features, it is verified that the highest mAP is obtained from deep features for most of the datasets because these features are automatically learned. The experimental results also show that using deep features with Hough forest performs the best. By employing Active Learning, it is demonstrated that after training around 2.5% of data, almost same performance as training on the whole dataset can be achieved. An annotation tool for user interaction during Active Learning is also developed.

The real-world dataset size can range from 1000 to 100000. State-of-the-art object detection methods are based on deep learning. In these cases, Active Learning can be incorporated as a preprocessing step to annotate only beneficial images from the entire dataset. This annotated dataset can be used with deep learning methods.

Future work

There are several options to explore in incorporating Active Learning methods.

The annotation cost is formulated assuming the cost to correct false positive detections is equal to cost to correct false negatives. In reality, this is not the case because the time taken in annotating false negatives is greater than annotating false positives. A user study in work [1] shows that effort to annotate false negatives is three times effort to annotate false positives.

Deep Learning methods can be used for end-to-end learning to obtain annotation cost [94].

Multiclass detection can be incorporated in Hough Forests by changing the split functions in classification and regression nodes for more than one class. In regression nodes, the offsets of multiple class should be minimised.

The uncertainty sampling-based methods suffer from the problem of learning noisy images. Therefore, other kinds of query methods can be explored.

The web-based annotation tool does not incorporate functionality for the user to correct false positives or add false negatives. Also, the functionality to see all the propagated annotations for the entire dataset is absent.

Bibliography

[1] A. Yao, J. Gall, C. Leistner, and L. Van Gool, "Interactive object detection," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3242–3249, IEEE, 2012.

[2] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach*, vol. 2. Prentice hall Upper Saddle River, 2003.

[3] J. O. Effoduh, "The fourth industrial revolution by klaus schwab," 2016.

[4] "AI is the new electricity." `https://medium.com/@Synced/artificial-intelligence-is-the-new-electricity-andrew-ng-cc132ea6264`. Accessed: 2018-03-02.

[5] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE multimedia*, vol. 19, no. 2, pp. 4–10, 2012.

[6] D. Murray and J. J. Little, "Using real-time stereo vision for mobile robot navigation," *autonomous robots*, vol. 8, no. 2, pp. 161–171, 2000.

[7] "Stanford CNN lecture." `http://cs231n.stanford.edu/syllabus.html`. Accessed: 2016-07-17.

[8] "HoloLens." `https://www.microsoft.com/en-us/hololens`. Accessed: 2016-07-17.

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

[10] "AmazonGo." `https://www.amazon.com/b?ie=UTF8&node=16008589011`. Accessed: 2016-07-17.

[11] "GoogleCar." `https://waymo.com/`. Accessed: 2016-07-17.

[12] "GoogleEcho." `https://store.google.com/product/google_home`. Accessed: 2016-07-17.

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

[14] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

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

[16] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, Dec. 2006.

[17] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 2879–2886, IEEE, 2012.

[18] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, pp. 2980–2988, IEEE, 2017.

[19] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3354–3361, IEEE, 2012.

[20] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, no. 1, pp. 21–71, 1998.

[21] R. M. Rangayyan, *Biomedical image analysis.* CRC press, 2004.

[22] H.-Y. Lee, J.-B. Huang, M. Singh, and M.-H. Yang, "Unsupervised representation learning by sorting sequences," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 667–676, IEEE, 2017.

[23] R. Bose, P. Buneman, and D. Ecklund, "Annotating scientific data: why it is important and why it is difficult," in *Proceedings of the 2006 UK e-Science all hands meeting*, vol. 23, pp. 121–148, Citeseer, 2006.

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[25] R. G. Cinbis, J. Verbeek, and C. Schmid, "Weakly supervised object localization with multi-fold multiple instance learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 1, pp. 189–203, 2017.

[26] H. Bilen, M. Pedersoli, and T. Tuytelaars, "Weakly supervised object detection with convex clustering," in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pp. 1081–1089, IEEE, 2015.

[27] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2, pp. II–II, IEEE, 2003.

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

[29] P. Vokuda, "Object classification and segmentation using rgb-d images," research and development thesis, HBRS, August 2017.

[30] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky, "Hough forests for object detection, tracking, and action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 11, pp. 2188–2202, 2011.

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

[32] B. Settles, "Active learning literature survey," Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[33] "Online annotation game." `http://www.gwap.com`. Accessed: 2018-02-14.

[34] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford, "Captcha: Using hard ai problems for security," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 294–311, Springer, 2003.

[35] "Mechanical Turk." `https://www.mturk.com/`. Accessed: 2018-02-14.

[36] "figure-eight." `https://www.figure-eight.com/platform/training-data/computer-vision/`. Accessed: 2016-07-17.

[37] "understand.ai." `https://understand.ai/`. Accessed: 2016-07-17.

[38] B. Settles, "Active learning literature survey. 2010," *Computer Sciences Technical Report*, vol. 1648.

[39] C. C. Aggarwal, X. Kong, Q. Gu, J. Han, and P. S. Yu, "Active learning: A survey," 2014.

[40] A. Krishnakumar, "Active learning literature survey," tech. rep., Technical Report, University of California, Santa Cruz, 2007.

[41] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman, "Discovering objects and their location in images," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1, pp. 370–377, IEEE, 2005.

[42] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 1, pp. I–I, IEEE, 2002.

[43] G. Rätsch, T. Onoda, and K.-R. Müller, "Soft margins for adaboost," *Machine learning*, vol. 42, no. 3, pp. 287–320, 2001.

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

[45] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[46] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[47] F. Baumann, A. Ehlers, K. Vogt, and B. Rosenhahn, "Cascaded random forest for fast object detection," in *Scandinavian Conference on Image Analysis*, pp. 131–142, Springer, 2013.

[48] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[49] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

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

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

[52] R. Girshick, "Fast r-cnn," *arXiv preprint arXiv:1504.08083*, 2015.

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

[54] D. P. Papadopoulos, J. R. Uijlings, F. Keller, and V. Ferrari, "Extreme clicking for efficient object annotation," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4940–4949, IEEE, 2017.

[55] D. P. Papadopoulos, A. D. Clarke, F. Keller, and V. Ferrari, "Training object class detectors from eye tracking data," in *European conference on computer vision*, pp. 361–376, Springer, 2014.

[56] D. P. Papadopoulos, J. R. Uijlings, F. Keller, and V. Ferrari, "Training object class detectors with click supervision," *arXiv preprint arXiv:1704.06189*, 2017.

[57] D. Lin, J. Dai, J. Jia, K. He, and J. Sun, "Scribblesup: Scribble-supervised convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3159–3167, 2016.

[58] H. Bilen, M. Pedersoli, and T. Tuytelaars, "Weakly supervised object detection with posterior regularization," in *Proceedings BMVC 2014*, pp. 1–12, 2014.

[59] D. P. Papadopoulos, J. R. Uijlings, F. Keller, and V. Ferrari, "We don't need no bounding-boxes: Training object class detectors using only human verification," *arXiv preprint arXiv:1602.08405*, 2016.

[60] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.

[61] I. Misra, A. Shrivastava, and M. Hebert, "Watch and learn: Semi-supervised learning of object detectors from videos," in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pp. 3593–3602, IEEE, 2015.

[62] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Advances in neural information processing systems*, pp. 529–536, 2005.

[63] G. Camps-Valls, T. V. B. Marsheva, and D. Zhou, "Semi-supervised graph-based hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 10, pp. 3044–3054, 2007.

[64] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation.," in *AISTATS*, pp. 57–64, Citeseer, 2005.

[65] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[66] M. Ristin, M. Guillaumin, J. Gall, and L. Van Gool, "Incremental learning of ncm forests for large-scale image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3654–3661, 2014.

[67] D. Jayaraman and K. Grauman, "Learning to look around: Intelligently exploring unseen environments for unknown tasks,"

[68] S. Argamon-Engelson and I. Dagan, "Committee-based sample selection for probabilistic classifiers," *Journal of Artificial Intelligence Research*, vol. 11, no. 335, p. 360, 1999.

[69] N. A. H. Mamitsuka *et al.*, "Query learning strategies using boosting and bagging," in *Machine learning: proceedings of the fifteenth international conference (ICML'98)*, vol. 1, Morgan Kaufmann Pub, 1998.

[70] Y. Zhao, C. Xu, and Y. Cao, "Research on query-by-committee method of active learning and application," in *International Conference on Advanced Data Mining and Applications*, pp. 985–991, Springer, 2006.

[71] A. Freytag, E. Rodner, and J. Denzler, "Selecting influential examples: Active learning with expected model output changes," in *European Conference on Computer Vision*, pp. 562–577, Springer, 2014.

[72] K. Konyushkova, R. Sznitman, and P. Fua, "Learning active learning from data," in *Advances in Neural Information Processing Systems*, pp. 4228–4238, 2017.

[73] T. Ross, D. Zimmerer, A. Vemuri, F. Isensee, S. Bodenstedt, F. Both, P. Kessler, M. Wagner, B. Müller, H. Kenngott, *et al.*, "Exploiting the potential of unlabeled endoscopic video data with self-supervised learning," *arXiv preprint arXiv:1711.09726*, 2017.

[74] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," *arXiv preprint arXiv:1703.02910*, 2017.

[75] J. Illingworth and J. Kittler, "A survey of the hough transform," *Computer vision, graphics, and image processing*, vol. 44, no. 1, pp. 87–116, 1988.

[76] "hough trnsform." `https://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm`. Accessed: 2016-07-17.

[77] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3, pp. 850–855, IEEE, 2006.

[78] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[79] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[80] A. Srikantha, "Characterizing objects in images using human context," doctor rerum naturalium, Rheinischen Friedrich–Wilhelms–University, Bonn, June 2017.

[81] M. Jambunathan *et al.*, "Some properties of beta and gamma distributions," *The annals of mathematical statistics*, vol. 25, no. 2, pp. 401–405, 1954.

[82] T. Kobayashi, A. Hidaka, and T. Kurita, "Selection of histograms of oriented gradients features for pedestrian detection," in *International Conference on Neural Information Processing*, pp. 598–607, Springer, 2007.

[83] M. B. Blaschko, "Branch and bound strategies for non-maximal suppression in object detection," in *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 385–398, Springer, 2011.

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

[85] A. Dutta, A. Gupta, and A. Zissermann, "Vgg image annotator via." http://www.robots.ox.ac.uk/ vgg/software/via/, 2016. Accessed: 2018-03-02.

[86] "Python." `https://www.python.org/`. Accessed: 2018-02-14.

[87] "Flask Web application." `http://flask.pocoo.org/`. Accessed: 2018-02-14.

[88] E. Borenstein and S. Ullman, "Combined top-down/bottom-up segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 30, no. 12, pp. 2109–2125, 2008.

[89] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results"." http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html".

[90] A. Steyer and Y. Schwab, "Fib-sem hela cervical cancer cell."

[91] C. Lassner and R. Lienhart, "The fertilized forests decision forest library," in *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 681–684, ACM, 2015.

[92] S. Aggarwal, *Flask Framework Cookbook.* Packt Publishing Ltd, 2014.

[93] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[94] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, pp. 1050–1059, 2016.

[95] "VGG19 network model." `https://lihan.me/2018/01/vgg19-caltech101-classification/`. Accessed: 2018-03-02.

Appendix A

Miscellaneous plots

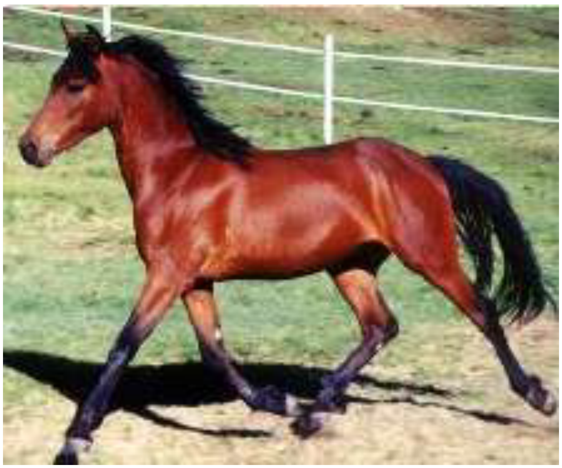


Figure A.1: Input image to visualize HoG features in 4.2.

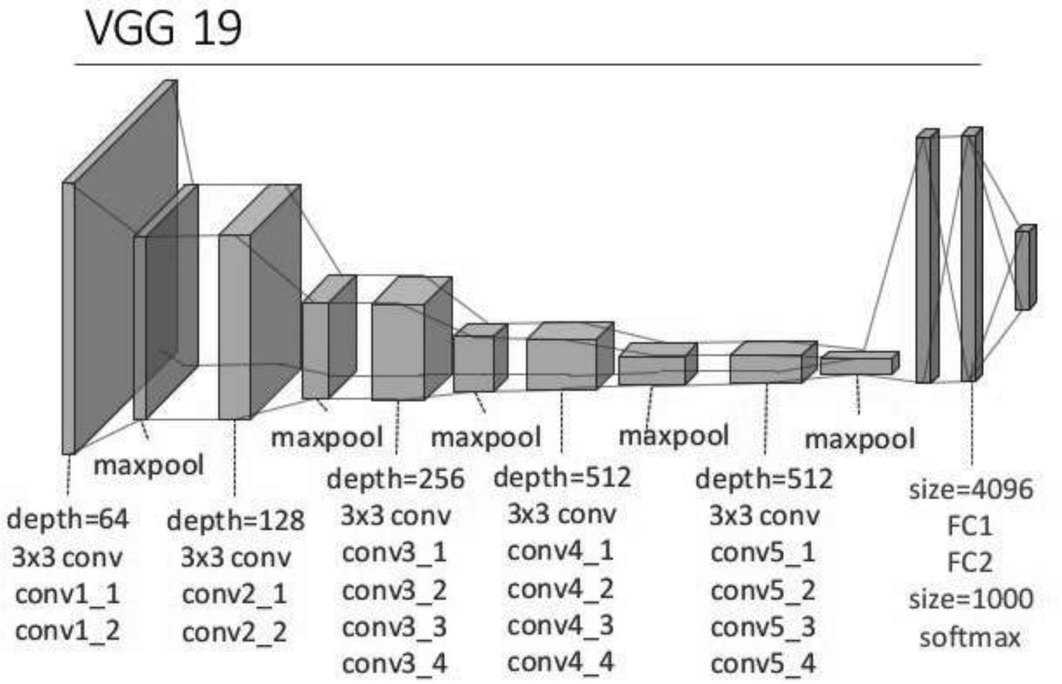


Figure A.2: VGG19 network architecture showing convolutional, pooling, fully-connected and softmax layers. Image from [95].