UNIVERSITY OF AMSTERDAM

MSc Artificial Intelligence
Master Thesis

Active Learning for Object Detection With Localization Uncertainty from Sampling-Based Probabilistic Bounding Boxes

by

Jasper Bakker

10260250

November 20, 2019

36 EC
January 2019 - November 2019

Supervisors:
Dr. Stevan Rudinac
Dr. Efstratios Gavves
Maarten Sukel

Assessor:
Dr. Pascal Mettes

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Abstract

Deep learning has spurred great advancements in object detection performance. However, to train deep neural networks, typically large annotated datasets are needed. Annotating images for object detection is tedious and costly. Active learning, in which the most informative images are annotated, is therefore a promising research direction to alleviate the annotation burden. Previous research into active learning for object detection mostly focused on classification uncertainty metrics, though some recent research showed improved performance when a localization component is added.

In this work we introduce a localization uncertainty metric based on advances in probabilistic object detection. In probabilistic object detection, the notion of rigid bounding boxes is replaced by a probabilistic one. This is done using two multivariate Gaussian distributions to denote the bounding box coordinates. We propose a novel sample selection strategy that utilizes the localization uncertainty obtained through the probabilistic bounding box. We scale this by the probability that an object is a foreground object and the size of the object. To obtain the probabilistic object detections, we use an ensemble of SSD detectors and perform extensive experiments with various techniques to merge the set of detections into probabilistic object detections. For the best performing merging strategy, pre-NMS averaging, we show promising results on 6 carefully chosen classes of the Pascal VOC 2007 dataset in both a multi-class as well as a two-class setup.

Furthermore, we combine the localization uncertainty with a density metric, which indicates whether an image is representative of the underlying data distribution. This shows us that the two less-performing merging strategies, BSAS merging and HDBSCAN merging, have a tendency to sample from low density regions of the image space. The pre-NMS averaging merging strategy does not display this behaviour.

Finally, for the best performing merging method we combine the localization uncertainty with a classification uncertainty metric. We show that, the classification uncertainty metric and the localization uncertainty are not complementary to each other. In most cases it is optimal to use only the proposed localization uncertainty metric to select the samples which should be annotated.

Aknowledgements

First of all I want to thank my supervisors Stevan, Stratis and Maarten. Stratis, even though we did not meet that often, I think the few meetings that we had were of great value to the direction of the thesis. I appreciate you have always been honest with me, even when I would not like it. Stevan, I want to thank you for always being ever optimistic, even when I was not at some points during the process. I think that, even though I learned a lot from you during our discussions about the subject, this optimism is what I learned most from. Maarten, you and everybody at the municipality were always so patient and gave me so much freedom. A typical dialog between us during a meeting would be "Does it work already Jasper?", "No, but I have a new plan...." I'm glad to finally be able to say that it works, even though I was already prepared to accept that it did not.

Then I would like to thank Dennis, Arend and Aron for proofreading my thesis and giving invaluable feedback and Oscar for allowing me to run (quite a few) experiments overnight in the last two weeks on his gaming PC. Without all of you it would never have been possible to write it like this.

I would also like to thank Aron specifically, for being the best study-buddy one can wish for in a master like this. We've spent quite a few weekends and even a few nights on Science Park (though I believe I beat our record now ;)). With you and of course Tim and Daan we even went to Montreal to present our paper, something I wouldn't have dreamed of when I was still studying Economics.

A big thank you goes to my family as well. You have always supported me during these years of studying. When I said that I wanted to change from Economics to something like Artificial Intelligence, you just wished that I would become happy with my choice and didn't care about the fact that it would take 3 (and a bit) more years to get there.

Finally, I want to thank Marisa. I know my life has been all about this master, especially since the thesis started. But you were always there for me, helping me and taking my mind of things when I needed it the most.

Thank you all! You guys are the best!

Contents

Chapter 1

Introduction

Human eyes capture about 10 gigabits of visual information each second of which the brain is capable of processing about 3 megabits in order to make sense of the world around us [2]. This process of obtaining a useful description of the external world from images is called vision [48]. Vision is the subject of research in many disciplines, from biology and psychology to computer science and artificial intelligence. Computer vision is the field that studies how machines can obtain the ability to see [76]. Enabling computers with the ability to see opens up a large variety of possibilities. This can be seen in the rise of applications such as medical image analysis, computational photography and self-driving cars, which heavily rely on computer vision techniques. In this thesis we focus on a fundamental computer vision technique called object detection. In object detection, the machine is required to classify the concepts it sees on an image as well as give their precise location [88]. Object detection is a challenging problem, as objects within a category can vary greatly in terms of appearance [19]. The appearance varies not only because of possible occlusion, changing viewpoints, or changing illumination, but also because of intra-class variability [19]. For example, a car looks completely different from the top than from the side or the front. It can also come in many different colors, and shapes. Moreover, as in any computer vision task, the algorithm must deal with very high dimensional input. For example, an image of 300 by 300 pixels with 3 color channels already has 270.000 input dimensions.

About a decade ago, most computer vision approaches focused on the use of hand-engineered features for image understanding [40]. In recent years, the use of deep Convolutional Neural Networks (CNNs) in combination with large annotated datasets have demonstrated a significant improvement in computer vision challenges [29, 71, 88]. These large annotated datasets are needed to train the substantial number of parameters that make up modern neural network architectures. However, manually annotating such datasets can be very time-intensive and therefore costly. For example, Russakovsky, Li and Li [65] found that verifying if a class C is on an image takes about 5.34 seconds, drawing a bounding box around an object about 10.21 seconds and naming what is in a bounding box 9.46 seconds. To put this into greater context, obtaining Microsoft's Common Objects in Context (MSCOCO) dataset took about 70.000 hours of annotation time [45]. Often crowd-sourcing is used to aid with the laborious annotation effort while keeping costs acceptable [54, 64, 45]. However, this is not always feasible. Confidential data, such as industrial or military data, or data requiring expert knowledge, such as medical images, cannot be annotated using crowd-sourcing [86]. To reduce annotation costs in such cases, an intuitive option is to find ways to reduce the amount of annotations needed.

To do this, many of the top-performing computer vision algorithms [46, 71, 62, 88] have relied on transfer learning. In transfer learning, the weights of a network trained on another dataset are transferred to one's own domain. Though very useful for learning visual features that generalize very well over various datasets [40], the need for an annotated dataset remains for any practical application. Even though visual features may generalize

well, the cross-dataset generalization of class definitions is far from perfect [77]. Moreover, more often than not, there is no annotated dataset available that contains the object categories one is interested in.

In this thesis we focus on a different strategy to reduce the amount of annotations needed, called *active learning*. Most machine learning research thinks of the learner as a passive receiver of the data it processes [15]. In active learning, it is hypothesized that by enabling the learning algorithm to choose the data it learns from itself, it will perform better with less training data [68]. Cohn, Ghahramani and Jordan [15] even show that for many machine learning algorithms, it is possible to compute a statistically 'optimal' way to select the training data. One of the most popular ways of active learning is to sample the data points the algorithm is most uncertain about [68]. Intuitively, this makes sense; if the algorithm already knows what a car looks like from the side, it might not learn a lot from another such image. However, if it is uncertain when a car is shown from the back, it might learn a lot more when such an image is added to the training data.

Object detection requires not only good classification of the objects present on an image, but also accurate localization. Therefore, we hypothesize that besides classification uncertainty, localization uncertainty can aid the active learning process. To this end we utilize probabilistic object detections. For probabilistic object detections, one needs both a distribution over the classes as well as over its location for each detection. Specifically, the bounding box location is defined by two multivariate Gaussian distributions [27]. We use the covariance matrices of these Gaussian distributions as source of localization uncertainty. We experiment with different methods of obtaining probabilistic object detections. As (classification) uncertainty sampling has been shown to have a risk of sampling outlier observations [68], we also combine the localization uncertainty with density sampling. In density sampling, one takes the relation of the sample to the rest of the dataset into account, in order to sample from high-density regions of the input space [68]. Moreover, we combine the localization uncertainty metric with a classification uncertainty metric as object detection consists of classification and localization. We observe that the two uncertainty metrics are not complementary, as in almost all settings the performance of the combination lies in between that of its components. Finally, because of the aforementioned risk of sampling from low-density areas of the input space we combine the localization uncertainty metric with the classification uncertainty metric and a density metric. In all experiments, we use an ensemble of Single Shot multibox Detector [46] object detectors (SSD). We use the SSD as it is one of the top object detection algorithms, both in terms of speed and detection performance. Especially the speed is important since we want to limit the computational cost, because these must weigh up to the decrease in labeling cost. We experiment on six carefully chosen object categories from one of the most popular object detection benchmarks, the Pascal VOC 2007 dataset [17], in a two-class setup and a multi-class setup. In both the two-class and multi-class cases we demonstrate promising results, as our localization uncertainty metric outperforms all other sampling strategies in the majority of the cases.

1.1 Research Question and Contributions

The research objective of this thesis is to use localization uncertainty to aid active learning for object detection. The main research question is the following:

Can localization uncertainty obtained with probabilistic object detection methods be used to improve active learning for object detection?

As obtaining localization uncertainty for active learning trough probabilistic object detection has not been attempted before, we divide the main hypothesis into the following sub-questions:

- What is the most beneficial way to obtain probabilistic object detections for the use of active learning?
- Does sampling with our localization uncertainty metric obtained trough probabilistic object detections suffer from sampling outliers (i.e. in low-density areas of the input distribution)?
- Does active learning with our localization uncertainty metric outperform sampling at random?
- Does active learning with our localization uncertainty metric outperform active learning with classification uncertainty?
- Does active learning with localization uncertainty outperform only sampling from high-density regions?
- Does the combination of localization uncertainty and classification uncertainty result in better performance than only using either one?

The contributions are the following:

1. We propose a novel localization uncertainty metric for active learning settings. We utilize the localization uncertainty obtained from the probabilistic bounding box and scale it by its size and probability of containing a foreground object. We show promising results in both a two-class as well as a multi-class setup.
2. We conduct extensive experiments with various ways of obtaining probabilistic object detection.
3. We propose various combinations of our localization uncertainty metric with classification and density sampling metrics. We show that the performance of these metric is in general worse than only using our localization uncertainty metric.
4. In an ablation study we show that the performance of the two scaling variables used in our localization uncertainty metric are beneficial for active learning performance.
5. Finally, we perform a qualitative analysis where we compare the distribution of difficult versus non-difficult objects for a few of our sample selection strategies. In two case studies we look at the actual images selected.

Chapter 2

Background

In this section we discuss the relevant background in both computer vision as well as active learning needed for the rest of the thesis. We start by shortly explaining what object detection is in relation to other computer vision tasks. After that, we continue by discussing object detection's most relevant architectures and techniques as well as the notion of probabilistic object detections. Finally, we discuss the background on active learning.

2.1 Recognition Tasks in Computer Vision

Within computer vision, the various recognition tasks can be roughly categorized as: *image classification*, *object localization*, *object detection*, *semantic segmentation*, and *instance segmentation*. Image classification is the task of identifying whether an object is in an image or not. Object localization is concerned with the task of identifying where on the image an object is, without predicting what it is. Object detection extends this by not only combining the two methods but also allowing multiple objects to be identified on an image, creating a bounding box for each instance of each object category of interest. Semantic segmentation moves away from bounding boxes and predicts for each pixel which class it belongs to, but does not make a distinction between different instances of an object category. Finally, instance segmentation determines for each pixel to what class it belongs, but also if it constitutes a different object instance. In figure 2.1, examples can be seen of most of the recognition tasks mentioned above. In this thesis we focus on object detection, which will be treated more in depth in the next section.

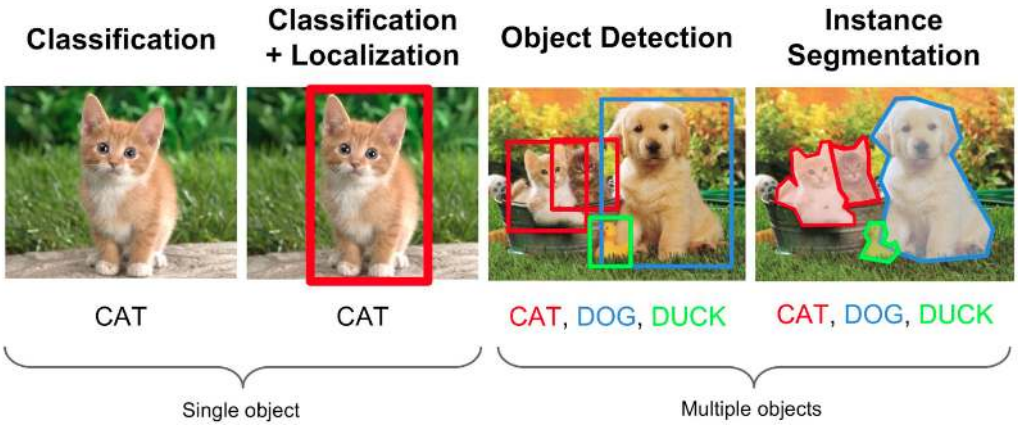


Figure 2.1: Examples of various recognition tasks in computer vision.

2.2 Object Detection

Object detection is the task of predicting *where* objects are located and to *what* class each object belongs to in a given image [88]. A detection can be formalized as $\boldsymbol{D_i} = \{\boldsymbol{c}_i, \boldsymbol{b}_i\}$, where $\boldsymbol{c}_i$ is a probability distribution over the classes and $\boldsymbol{b}_i$ are the bounding box coordinates. Until a few years back, the most successful approach to object detection was the *sliding window approach* [34], also referred to as *exhaustive search* [78]. In this approach, the whole image is scanned using sliding windows at multiple scales. Then, a computationally efficient, binary classifier is used for each class to determine whether an object of that class was present in that window [34, 88, 10]. At a single scale, the sliding window approach already generates around $10^4 - 10^5$ windows per image [34] making it computationally very expensive. Current methods decrease the classifiable object windows drastically by generating a smaller amount of well chosen bounding boxes and refining those, allowing the use of computationally more expensive classifiers [34, 88] such as Convolutional Neural Network (CNN) based classifiers.

In the remainder of this section we will first discuss Non-Maximum Suppression (NMS), which is an important technique that aims at selecting the best bounding box for an object out of multiple overlapping ones. Afterwards, we discuss important works in the development of CNN-based object detectors and the key techniques that are used in them. Then we discuss (mean) Average Precision (mAP), which is the most commonly used evaluation metric in object detection. Finally, we will discuss probabilistic object detection, which deviates from the formulation of rigid bounding box predictions.

2.2.1 Non-Maximum Suppression

Often a single object is detected in multiple neighbouring bounding boxes. Therefore, a form of *Non-Maximum Suppression* (NMS) is used in most object detection algorithms to select one of them and suppress others. The standard for NMS in object detection is greedy NMS [88]. For each class, NMS starts with a list of bounding boxes $\mathcal{B}$ and a list of confidence scores $\mathcal{S}$ [10]. The detection $\boldsymbol{D}_i$ with the highest confidence score $\mathcal{M}$ is then selected by removing its bounding box and confidence score from $\mathcal{B}$ and $\mathcal{S}$ and appending it to the final set of detections $\mathcal{D}$. In greedy NMS, the bounding boxes that have a higher spatial similarity with $\boldsymbol{D}_i$ than some threshold σ are also removed from $\mathcal{B}$ and $\mathcal{S}$. This spatial similarity is measured by the *intersection over union (IoU)*, which is the area of overlap divided by the combined area of the boxes. This continues as long as $\mathcal{B}$ is not empty. An issue of greedy NMS is that it completely removes the bounding boxes with a IoU above σ [10]. If an object would have been present in the bounding box, this could hurt performance. In order to alleviate this issue, a different method, called soft-NMS, has been proposed in which bounding boxes are not removed. Instead their confidence score is weighted by the IoU score with the selected bounding box [10].

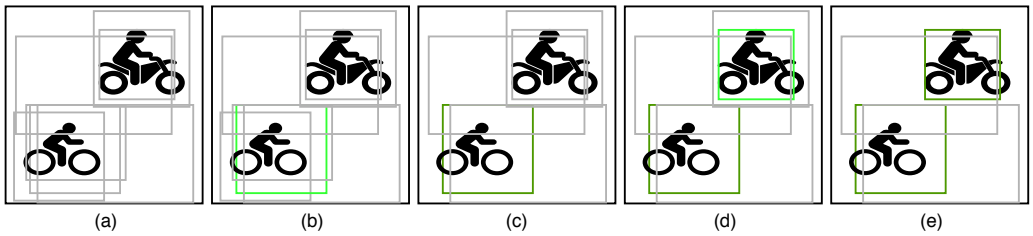


Figure 2.2: Greedy Non Maximum Suppression (NMS): a) a set of overlapping detections b) the highest confidence detection is selected (light green) c) for the selected detections the IoU with the other detections is calculated. Detections with an IoU higher than a threshold are suppressed d) the highest remaining confidence detection is selected e) again the detections with a high enough IoU are suppressed. Note that not all bad (grey) detections are necessarily suppressed though.

2.2.2 CNN Based Object Detectors

Girshick et al. were the first to successfully propose a deep learning architecture for object detection, called the Regions with Convolutional Network features (R-CNN) [24]. It was the first of the deep learning based *two staged object detectors*, also commonly referred to as *region proposal-based frameworks* [88]. It considerably improved performance compared to the deformable part model [19], which was the state-of-the-art at the moment [88]. It first uses selective search [78] to obtain a set of about 2000 class-agnostic region proposals. It then uses a CNN, pre-trained on ImageNet [64], to extract features. For each class, it classifies each bounding box to be background or the class of interest using a linear Support Vector Machine (SVM). Greedy NMS is then applied to the bounding boxes.

Finally, a class-specific *bounding box regression* is applied to refine the localization of the objects. In Bounding box regression, a regression model is learned to improve the 4 coordinates of the proposed bounding box using the image features of that proposed box. In the case of R-CNN, they used ridge regression [24]. Bounding box regression is an important technique in many CNN-based object detectors [24, 23, 62, 46, 61]. A schematic overview of R-CNN can be found in Figure 2.3.

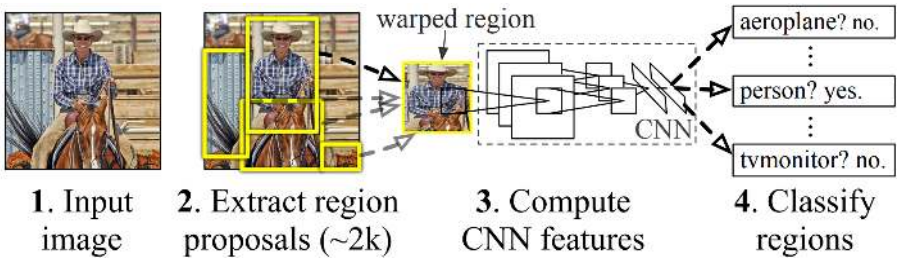


Figure 2.3: Flowchart of the R-CNN framework. (1) It first takes an input image (2), extracts around 2000 region proposals, (3) computes the CNN features (4) and uses class-specific linear SVMs to classify each region. Taken from [24]

.

Though R-CNN improved greatly upon the state-of-the-art, there were some disadvantages. Training is expensive in both disk space and time as it is a multi-stage pipeline: first, the CNN is fine-tuned on object proposals, then the softmax classifier is replaced by an SVM and fine-tuned and finally the bounding box regressor is trained, where for each region proposal the features are stored on disk [88]. Moreover, the selective search procedure is time-consuming and due to the existence of fully connected layers in the CNN, it requires a fixed size input.

Later, Girshick proposed Fast R-CNN [23], which improved upon R-CNN in multiple ways. In Fast R-CNN, the input is still an image and a set of region proposals obtained with selective search [78]. However, following the spatial pyramid structure of SPP-net [28], it shares computations between the region proposals. Specifically, it does this by first generating feature maps through a CNN before using the region proposals. Therefore, Fast R-CNN performs no redundant computations between overlapping region proposals. Another advantage is that all layers can be trained end-to-end. Fast R-CNN achieves this by using a Region of Interest (RoI) pooling layer. The RoI pooling layer divides the $h \times w$ RoI window into an $H \times W$ grid of subwindows and then uses max-pooling per subwindow to obtain a fixed size $H \times W$ output. This is then passed through fully connected layers to the two heads of the model, one for classification and one for predicting per-class bounding box regression offsets, after which NMS is applied. Finally, a weighted sum of the classification and localization loss, first introduced in Multibox [16], is used. This jointly optimizes the classification and the bounding box regression allowing the network to be trained end-to-end. The architecture of the Fast R-CNN network can be found in Figure 2.4.

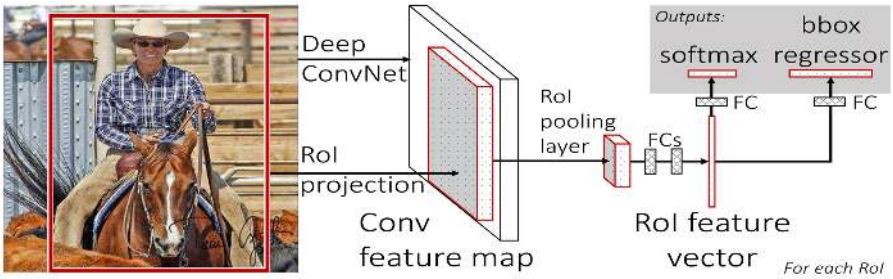


Figure 2.4: Architecture of the Fast R-CNN network. The fully convolutional network takes as input an image and multiple Regions of Interest (RoI). The resulting features for each RoI are put trough an RoI pooling layer and then mapped into a feature vector by fully connected (FC) layers. Finally, the network has two heads, one for classification and one for predicting per-class bounding box regression offsets. Taken from [23].

Finally, Ren et al. [62] proposed Faster R-CNN. In Faster R-CNN, the selective search algorithm [78] is replaced with a Region Proposal Network (RPN) to generate regions of interest. By sharing the convolutional layers with the detection network, the efficiency is increased. It makes the architecture a lot faster and truly end-to-end, only requiring an image as input. The RPN is a fully convolutional network allowing the input to be of arbitrary size. It generates the object proposals by sliding a window over the final convolutional feature map. For each location, the RPN predicts a *objectness score*, which measures membership to a set of object categories versus the background, and performs bounding box regression on a set of *anchor boxes* Figure 2.5. Anchor boxes are fixed bounding boxes of different scales and aspect ratios. The anchor boxes in the default implementation are a combination of 3 different aspect ratios $(1:1, 2:1, 1:2)$ and 3 different scales $(128 \times 128, 256 \times 256, 512 \times 512)$. At each location, this results in $k = 9$ boxes on which the RPN predicts foreground and background. For a convolutional feature map of $W \times H$ (typically about 2400), this results in $W \times H \times k$ anchors. Finally NMS is applied, typically leaving about 2000 proposals.

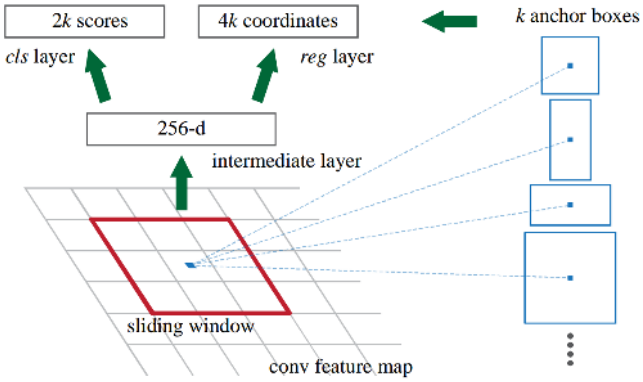


Figure 2.5: Region Proposal Network (RPN). A sliding window is used over the final convolutional feature map. At each location for a set of k anchor boxes, it predicts bounding box regression offsets and objectness scores. Taken from [62]

The RPN is only used for region proposal generation. For the object detection, a Fast R-CNN fed with the RPNs object proposals is used. However, both training RPN and Fast R-CNN independently will result in different convolutional layers. Therefore Ren et al. [62] propose three methods of alternating the training between the two while allowing a shared CNN backbone. Firstly, they propose to alternate training of the two networks, which is done in all their experiments. Secondly, they propose to approximate joint training where the the region proposals for the Fast R-CNN are treated as fixed proposals. Thirdly, they propose non-approximate joint training, where the backpropagation is done all the way. This is done by using *RoI warping* [62], which allows for backpropagation trough the otherwise non-differentiable RoI pooling layer. A schematic overview of the Faster R-CNN architecture can be found in Figure 2.6.

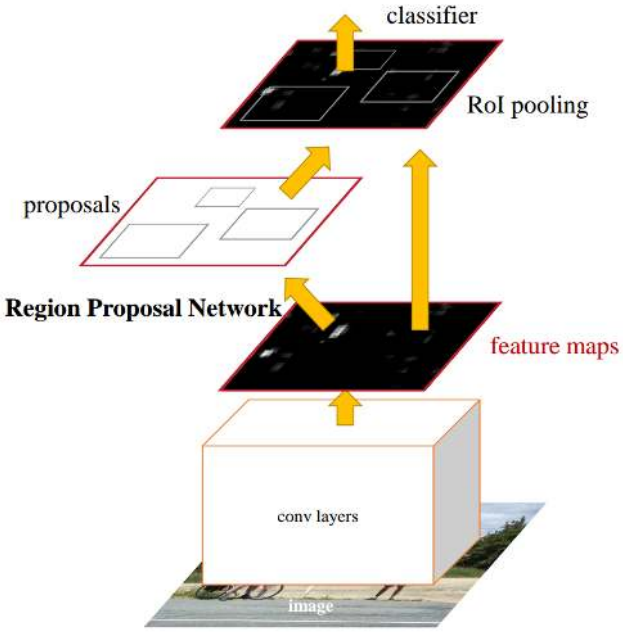


Figure 2.6: Flow chart of the Faster R-CNN architecture. The network has a region proposal network (RPN) which serves as 'attention' for the classifier. Taken from [62].

The previous methods are called *region proposal-based frameworks* [88], as they require to first generate region proposals on which they apply their regression and classification heads. Despite their success, they were still not fast enough to run in real-time because of their two-stage approach. A different approach is taken in the *Regression/classification-based works* [88], also commonly referred to as *one-stage object detectors* [52, 88]. In one stage detectors, the bounding boxes and class probabilities are both directly mapped from the image pixels [88]. The detection time is typically a lot faster than for region proposal-based frameworks. An important work is the You Only Look Once (YOLO) approach by Redmond et al. [61]. In YOLO, the image is first resized to a fixed size and then split into an $S \times S$ grid. A grid cell is responsible for detecting an object if its center lies in that grid cell. Each grid cell is split into B bounding boxes for which *objectness* confidence scores are predicted. At the same time, conditional class probabilities are calculated for each grid cell: $p(class_i|object)$, where the class probability is conditioned on the grid cell containing an object. YOLO does not perform well for small objects in groups. This is caused by the strong spatial constraints on bounding box predictions, namely that each grid cell only predicts a fixed number of B boxes and can only have one class. Moreover, YOLO treats errors for small bounding boxes the same as for larger bounding boxes, while a small error in a small bounding box leads to a much greater effect on IoU than a small error on a large bounding box. Finally, YOLO does not generalize well to objects in new aspect ratios and uses coarse features to predict bounding boxes because of multiple downsampling layers present in the architecture. YOLO is a lot faster than Faster R-CNN (45 Frames Per Second (FPS) vs. 7 FPS), but this speed comes with a substantial decrease in detection accuracy. The main idea behind the YOLO framework can be found in Figure 2.7.

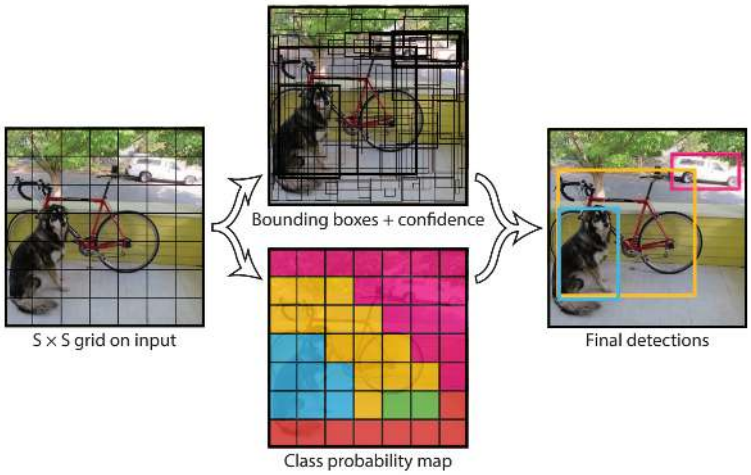


Figure 2.7: The main idea behind YOLO. It generates an $S \times S$ grid on the input. Each grid cell detects bounding boxes + objectness scores (top) and classification probabilities (bottom), which are then combined into detections [61].

The Single Shot multibox Detector (SSD) [46] aims to alleviate the weaknesses of these methods and achieve a high-accuracy object detector with real-time detection speed. It predicts at 59 FPS and, though using a more involved data augmentation scheme, reports a slightly better performance than Faster R-CNN [46]. The standard SSD300 framework uses a VGG-16 [71] base network pre-trained on ImageNet [64], without the fully-connected layers, and adds 4 convolutional layers to this base network. Instead of fixed grid cells as in YOLO, the SSD utilizes default boxes, which are similar to the anchor boxes in Faster R-CNN, but are applied to feature maps at different scales to help handling different object scales. They do this by using convolutional filters of different layers in the network. Moreover, they start from pre-calculated prior-boxes, and use bounding box regression to learn to predict the final bounding boxes. The bounding boxes are class-agnostic and for each bounding box a class probability distribution is calculated. The final predictions are obtained by efficiently running greedy NMS, as NMS is only performed on a box if the class probability is higher than some confidence threshold. The network is trained using a weighted sum of localization loss (Smooth L1) and confidence loss (Softmax). As the SSD is the object detection framework used in this thesis, it will be treated in greater detail in section 4.1.

Newer object detectors [88] and tricks like other ways of applying NMS [10, 31] or adjustments to the loss functions [31, 44] exist. However, in this thesis we do not aim to improve existing detection frameworks but aim to improve the sample selection process for object detection, making these methods out of scope for the thesis.

2.2.3 Mean Average Precision (mAP)

The evaluation metric most often used for object detection is mean average precision (mAP). To evaluate the localization performance of an object detector, the set of pixels in the predicted bounding box A must be compared with the set of pixels in the ground truth bounding box G. This is commonly done by comparing the IoU of the sets of pixels of the two bounding boxes:

$$IoU(A, G) = \frac{|A \cap G|}{|A \cup G|} \tag{2.1}$$

This metric is larger when the bounding boxes have a large overlap, and smaller when there are pixels that do

not overlap. For example, G can be completely subsumed by A, but if A is too large, the IoU will still be low. The higher the IoU, the more the prediction corresponds to the ground truth, where an IoU of 1 means that they are exactly the same.

A prediction is considered a True Positive (TP) if the IoU is higher than some threshold. The prediction is considered a False Positive (FP) if the IoU is below the threshold. The prediction is considered a False Negative (FN) if the ground truth is not detected. True Negatives (TN) would be all possible bounding boxes that were correctly not detected, but this is generally not considered. These values are calculated per class. One of the most used benchmark datasets, Pascal VOC, uses a threshold of 0.5 [17]. They set the threshold deliberately not too high to account for inaccuracies for objects that are highly non convex, such as a person that has its arms and legs spread [17].

Following the Pascal VOC challenges [17] , if a single ground truth overlaps with multiple predictions, the prediction with the highest IoU is considered a correct prediction (TP) and the rest is considered a false prediction (FP). All detections on all images are then sorted on their class confidence and the accumulative TPs and FPs are calculated from the most confident detection to the least confident detection.

These values are then used to calculate the precision and the recall:

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

Average Precision (AP) is calculated by approximating the Area under the Precision Recall curve. In the early Pascal VOC challenges this was using an 11-point interpolation method [17]:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p_{interpolated}(r) \tag{2.4}$$

where $p_{\text{interpolated}}(\text{r}) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r})$ and $p(\tilde{r})$ is the precision at recall $\tilde{r}$. However, since 2010 the Pascal VOC challenge moved towards a numerical integration of the precision-recall curve, effectively sampling all unique values on the curve. The idea behind this is to have a more precise metric and to be able to measure the difference for methods with low AP.

Finally, the mean Average Precision (mAP) can be calculated by taking the arithmetic mean of the APs over all classes:

$$mAP = \frac{1}{|classes|} \sum_{c \in classes} AP_c \tag{2.5}$$

The mAP is used for most popular object detection benchmarks, including PASCAL VOC [17] and MS COCO [45]. Critiques on the use of AP exist. Most notable is the fact that Average Precision does not explicitly include localization accuracy [55]. In the MSCOCO object detection challenge [45], a slightly different definition is used

in order to alleviate this. Instead of using a fixed IoU threshold, such as 0.5 for Pascal VOC [17], they take the mean AP values over a range of thresholds.

2.3 Probabilistic Object Detection

In the regular object detection task, the detection algorithm is required to predict a bounding box and a probability per class for that bounding box. In probabilistic object detection, as described in the work of Hall et al. [27], the following is required for each predicted object on the image:

- A categorical distribution over all class labels
- A bounding box, represented as $\mathcal{B} = (\mathcal{N}_0, \mathcal{N}_1) = (\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0), \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1))$, s.t. $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the mean and covariances for a multivariate Gaussian distribution describing the top-left and bottom-right corner of the box, respectively.

The multivariate Gaussian distributions at the corners can be integrated to obtain a probability per pixel of it belonging to the bounding box. Hall et al. [27] mention robotics applications and self driving cars as cases in which meaningful probabilistic object detections can be useful. In Figure 2.8 a visualization of probabilistic object detection can be found.

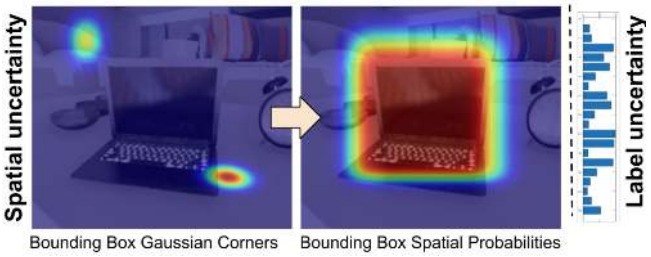


Figure 2.8: Probabilistic object detection does not only express semantic uncertainty, but also spatial uncertainty. Taken from [27].

Typically, probabilistic object detections are obtained through sampling-based uncertainty techniques [52, 51] such as Monte Carlo (MC) dropout networks [21] or an ensemble of neural networks which is referred to as a *deep ensemble* [42]. In the rest of this work we follow Miller et al. [52, 51], who refer to these methods, when applied to object detection, as *sampling-based object detectors*.

2.3.1 MC Dropout

Dropout is a regularization method where neurons are randomly dropped during training, first proposed by Hinton [32] and further researched by Srivastana et al. [75]. It is used in many Deep Learning models to avoid this overfitting and provides a way of approximately combining exponentially many different neural networks in an efficient manner. [75]. In the simplest case, each neuron has a fixed probability p to be turned off, often set at 0.5 for hidden neurons, but closer to 1 for input neurons [75]. In this way, a neural network with n units, can be seen as a possible collection of 2^n *'thinned'* networks. By applying this dropout to each training case, a different thinned network is trained (almost) every time, while the used weights in the different networks are shared between them. At test time, the dropout is normally not applied. The output of each neuron is then scaled by it's dropout probability p. Therefore, the expected output of the neuron is the same as during training time, but there is no need to take an explicit average over the exponentially many thinned models. During testing in Monte Carlo (MC) dropout networks, multiple stochastic forward passes through a network with dropout layers are performed and the results are averaged [21]. Gal suggests that using MC dropout results in

approximate Bayesian interpretation of neural networks [21]. Osband critizes this notion as he states that Gal confuses uncertainty with empirical risk [56]. Nevertheless, it is a way of estimating predictive uncertainty [42], and it can be used to obtain to obtain probabilistic object detections [50, 51].

2.3.2 Deep Ensembles

Inspired by MC dropout, which can be interpreted as an ensemble of neural networks, Lakshminarayanan, Pritzel and Blundell propose an alternative way of obtaining predictive uncertainty [42]. They propose to train multiple neural networks separately and average the predictions afterwards. To ensure sufficient stochasticity, They empirically found that for good predictive uncertainty it is sufficient to use a different order in which the samples are shown and a different random initialization for all the networks in the ensemble. Though this method has no Bayesian interpretation, it demonstrates better predictive uncertainty than MC dropout networks in many tasks [42], even though a significantly lower number of forward passes is performed. An additional benefit is the fact that the networks can be efficiently trained using a distributed environment.

2.4 Active Learning

In machine learning it is common to see the learning algorithm as a passive receiver of the data [15]. In active learning, the model is trained in an iterative manner [83, 68]. The key hypothesis is that by letting the learning algorithm choose the data from which it learns, it performs better with less training data [68].

There are three main categories of active learning: membership query synthesis, stream-based active learning and pool-based active learning [83]. In membership query synthesis, the learning algorithm can request labels for any unlabeled instance in the input space, including those it generates itself [68]. An example of this is the work by Huijser and van Gemert [37] in which the use deep generative models to generate new instances on the decision boundary. The main assumption in stream-based active learning is that obtaining an unannotated sample is (as good as) free. It is called stream-based or sequential active learning because typically the instances are shown one at a time and the learning algorithm must decide each time whether to request its label or to discard it. This is typically used in online learning settings [14, 4]. In this thesis we will focus on pool-based active learning, in which we have a small initial set, or *seed* set, of annotated and a large pool of unannotated data. The pool of unannotated data is evaluated entirely and ranked in order to determine the best samples to be annotated [68]. There are typically two parts in an active learning system, a learning algorithm and a sample selection method, working in an iterative manner. First the learned model makes predictions on the samples in the unlabeled pool. Then the sample selection algorithm selects the samples that would be most informative. An oracle, usually a human, annotates these samples. After this, the model is retrained on the annotated dataset, which is now extended by the newly annotated data. This model is then again used to make predictions on the, now slightly smaller, pool of unannotated data. An illustration of this process can be found in Figure 2.9.

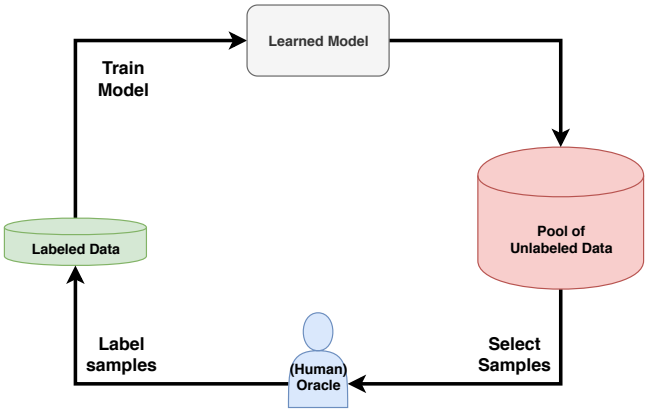


Figure 2.9: Diagram of a generic pool-based active learning framework [68].

2.4.1 Sample Selection Strategies

The key component in an active learning framework is how to determine the "informativeness" of a sample [85], often referred to as the *sample selection strategy*, *sample selection method* or *query strategy* [80, 82, 85]. In this section we will evaluate the sample selection strategies most important for this thesis. In this section we will refer to $\boldsymbol{x}^*$ as the 'most informative sample'.

The most common sampling framework is *uncertainty sampling*, which samples the data points the model is most 'uncertain' about [68]. Different types of uncertainty metrics exist, well known examples are *least confidence* sampling and *entropy* [68]. Least confidence sampling is formulated as follows:

$$\boldsymbol{x}^*_{LC} = \arg\max_{\boldsymbol{x}} 1 - p_\theta(\hat{y}|\boldsymbol{x}) \tag{2.6}$$

where θ are the model parameters and $\hat{y} = \arg\max_y 1-(p_\theta(y|x))$. In the case of binary classification, this would be the sample whose posterior distribution would be nearest to 0.5 [68]. Entropy is a measure that represents the amount of 'information' that is needed to encode a probability distribution [68]. In machine learning it is often used as an uncertainty measure and for active learning it can be formulated as follows:

$$\boldsymbol{x}^*_H = \arg\max_{\boldsymbol{x}} -\sum_i p_\theta(y_i|\boldsymbol{x}) \log p_\theta(y_i|\boldsymbol{x}) \tag{2.7}$$

An important drawback of uncertainty sampling is that it focuses on individual instances, rather than on the entire input space. Therefore, these methods are at risk of sampling outliers [68]. An illustration of when uncertainty sampling can be a poor strategy can be found in Figure 2.10, where the least certain instance lies on the classification boundary, but is unlikely to improve the accuracy as it is an outlier.

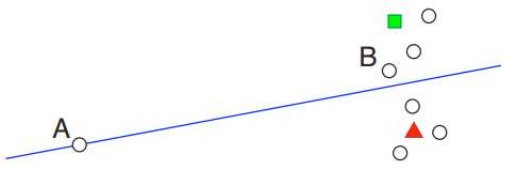


Figure 2.10: Point A, on the decision boundary, is the most uncertain. However, point B is likely to result in more information regarding the data distribution as a whole. Taken from [68].

Other methods that do focus on the entire input space exists, examples are sampling methods based on *Expected error reduction*, *Variance reduction* and *Density weighting* [68]. For this thesis we focus on density-based techniques. The idea is that informative samples are not only uncertain samples, but also *'representative'* of the underlying input distribution [68]. An example of density weighting-based sampling methods are clustering methods. Nguyen and Smeulders [53] suggest that clustering information can be useful in two ways. Firstly, cluster centers are considered representative samples which should be selected first. Secondly, samples located in the same cluster are likely to have the same label.

Another way of using density sampling is by assigning representativeness scores. This can then be easily combined with uncertainty scores, with the use of the *information density* metric [69]:

$$\boldsymbol{x}_{ID}^{*} = \text{argmax}_{\boldsymbol{x}} \left((\phi(\boldsymbol{x}))^{1-\beta} \cdot \left(\frac{1}{U} \sum_{u=1}^{U} sim(\boldsymbol{x}, \boldsymbol{x}^{u}) \right)^{\beta} \right) \tag{2.8}$$

where ϕ can be an uncertainty metric, $sim(\boldsymbol{x}, \boldsymbol{x}^{u})$ is the similarity between two samples according to some similarity metric and β is a weighting parameter. In his review, Settles found that methods that incorporate density sampling are often superior to methods that only consider uncertainty sampling [68].

A slightly different way of using an interactive setup is *user relevance feedback* [83]. In user relevance feedback the user is asked to annotate whether instances are 'relevant' or 'irrelevant' according to whether they can be associated with the given concept [36]. These methods are often used in multi-label image or video annotation and retrieval tasks [83] or for example on very large datasets [87]. Uncertainty-based methods can also be used in such cases, but directly sampling the images that have a high probability to be relevant is often more effective [83]. Wang et al. attribute this to the fact that positive samples are usually less frequent than negative samples in these tasks, as well as the fact that often ranking metrics such as Average Precision are used for evaluation [83].

Many other sampling strategies exist, for more extensive reviews the reader is referred to Wang and Hua [83] and Settles [68]. An important remark made by Wang and Hua [83] is that all of the above mentioned methods assume that the cost of annotating a sample is uniform, meaning that for a human annotator every sample will take approximately the same amount of time and/or all samples are annotatable by the same person, i.e. there is no difference in the amount of (expensive) expert knowledge that is needed per sample. However, this is not the case for many applications of active learning.

2.4.2 Definition of a Sample

In active learning, the goal is to select the most informative samples. However, the definition of a sample for object detection is not straightforward. An intuitive definition might be to see objects as samples, from hereon referred to as *object-level samples*. If there are multiple objects corresponding to the class of interest on an image, but only one of them is highly informative, it might make sense to only annotate this object. This could save a lot of time for the annotator. However, two potential problems arise from this definition. Firstly, it might be hard for the annotator to know which object to annotate, if the predicted bounding box overlaps with multiple objects of interest. Secondly, one ends up with possibly confusing, partially labeled images. Objects of interest that are present on images, but are not yet labeled, will considered to be background. If not handled correctly this could deteriorate the performance of the object detector. As for such images the training loss is higher when the detector would correctly detect an unannotated object of interest. See Figure 2.11 for an example of these two issues.

(a) (b)

Figure 2.11: Problems with object level labeling. A) Should the human annotator label the horse, the person on the horse or the boy right next to the horse? B) When training on the complete image where only the person on the horse is annotated, the rest is regarded as background. Raw image from Pascal VOC 2007 [17].

Another definition might be to define samples on the image level, from hereon referred to as *image-level samples*. The most informative image could then be considered as the one with the most informative object or the one with the highest summed or averaged informativeness of all the objects present on the image. The problem with this approach is that many uninformative objects might be annotated, allocating the annotation resources sub-optimally. In this thesis we employ the definition of image level samples.

Chapter 3

Related Work

This research lies on the intersection of the fields of object detection and active learning. However, much of the research on active learning within computer vision focuses on classification and segmentation tasks. As we employ an image-level sample definition, we find research in this field to be relevant for this work as well. We start by discussing the works in active learning for other computer vision tasks. Then we discuss the relevant work on active learning using object detection. Finally, we discuss the related work on how to obtain probabilistic object detections.

3.1 Active Learning for Computer Vision

Many studies have been conducted regarding active learning for other computer vision tasks. Compared to our research, the work discussed below differs in which computer vision task is experimented on, but is relevant in terms of their sample selection criteria or the use of sampling-based models.

3.1.1 Uncertainty-Based Active Learning for Image Classification

Neural networks are typically overconfident in their predictions [42]. Sampling-based models can be used to improve the predictive uncertainty [42, 22, 8]. Gal, Islam and Ghahramani [22] evaluate different uncertainty-based sample selection methods for both deterministic CNNs with dropout as well as Bayesian CNNs using MC dropout. They show that the sample selection methods all work better using Bayesian CNNs and that their best sample selection criterion, variation-ratios, needs about 3 times less data than random sample selection. Using the same uncertainty metrics, Beluch et al. [8] found that by using uncertainty estimates from an ensemble of CNNs he could strongly outperform the Bayesian CNNs that used MC dropout on multiple datasets. Moreover, they needed about a third less data than the MC dropout variant. They expect this to be because the MC dropout case has a fixed initialization for all its forward passes, and a fixed order of training batches whereas the ensemble method does not. Moreover, the total model capacity of the networks is larger for the ensemble methods than for the MC dropout case.

3.1.2 Active Learning in Computer Vision with Density and Uncertainty Sampling

Different ways of combining density sampling and uncertainty sampling have been tried for active learning in computer vision. In this work, the goal is to combine our localization uncertainty metric (that will be

described in section 4.3), with a density sampling metric. This is done in order to alleviate the potential problem of selecting outliers, while preserving its informativeness. No method has combined density sampling with uncertainty sampling for active learning in object detection.

The work of Li et al. [43] used the *information density* metric, earlier proposed in an Natural Language Processing (NLP) setting [69] for image classification. They use the information density metric first proposed by Settles and Craven [69]: $h_\beta(\boldsymbol{x}_i) = f(\boldsymbol{x}_i)^\beta d(\boldsymbol{x}_i)^{1-\beta}$ where $f(\boldsymbol{x}_i)$ is the conditional entropy over classes and β is a parameter to adjust the weighing of density and uncertainty. As finding the proper β is an important issue Li et al. [43] proposed a method of adaptively setting the β parameter over the active learning iterations. In short, for a range of β values (e.g. $[0.1, 0.2, ..., 0.9, 1]$), they select the set S of optimal samples $\boldsymbol{x}_i$ for each β value using $h_\beta(\boldsymbol{x}_i)$. They only annotate a single sample at a time, so the optimal β then equates to finding the optimal $\boldsymbol{x}_i$. They do this by minimizing the expected loss for each $\boldsymbol{x}$. For all $\boldsymbol{x}_i$, they compute the label probability for each possible label $p(y|\boldsymbol{x}, \theta_L)$, where θ_L is the current classifier trained on the current labeled dataset L. They add each possible instance-label pair $\langle \boldsymbol{x}, y \rangle$ to L and retrain the classifier on the augmented labeled set. The optimal $\boldsymbol{x}_i^*$ is then selected using the following equation:

$$\boldsymbol{x}_i^* = \arg\max_{\boldsymbol{x} \in S} \sum_{y \in \mathcal{Y}} p(y|\boldsymbol{x}, \theta_L) \left(\sum_{i \in U} (1 - p(\hat{y}_i | \boldsymbol{x}_i, \theta_{L+\langle \boldsymbol{x}, y \rangle})) \right)$$

where $\theta_{L+\langle \boldsymbol{x}, y \rangle}$ denotes the new model parameters after retraining the classifier on $L + \langle \boldsymbol{x}, y \rangle$, and $\hat{y}_i$ is the predicted label for $\boldsymbol{x}_i$. As can be seen, their method of adaptively setting the β parameter is both computationally expensive and only considers expected classification loss. In object detection, one would ideally also include localization loss, which can't be added in a similarly intuitive way. They tested their method on a range of image classification datasets. Their adaptive setting of β strongly outperformed other methods as well as sampling with a fixed β. However, they also found that without the adaptive adjustment of β, their sample selection strategy could sometimes perform worse than random sample selection.

Yang et al. [84] propose an active learning framework for biomedical image segmentation. They use their own Fully Convolutional Neural Network (FCN) architecture and apply this on the MICCAI Gland Challenge [72] and the ISBI EM Challenge [3]. To find the most important annotation areas, they combine uncertainty sampling with density sampling. To obtain uncertainty metrics they use bootstrapping: training a set of models, each on a subset of the data, sampled with replacement. They calculate the variance of the per pixel prediction among these models. Furthermore, they state that as deep learning models tend to be uncertain for similar types of instances, just the use of uncertainty would result in duplicated selections of annotation areas. Therefore, they also select representative samples that are highly similar to other samples. To do this, they calculate the cosine similarity between images, by utilizing the encoding provided by their neural network. They use a sequential way of combining their metrics: first selecting the subset of the images with the highest uncertainty, and then greedily selecting a smaller subset in which the images have the highest summed similarity score with all other images in the full dataset. Their density metric is thus similar to the density part in *information density*, only the similarity score is taken over the full dataset and not just the unlabeled part. In their experiments, they show that their sample selection method is more efficient than randomly sampling or just using the uncertainty criterion.

3.1.3 Image Representations

In order to compare the semantic distance of images in a meaningful way, a lower dimensional representation and a suitable distance metric is needed. A lot of research regarding this has been done in the field of Content Based Image Retrieval (CBIR), specifically for Query By Example challenges. In QBE, a query image is used

and similar images should be retrieved from the database of unlabeled images. For example in the Paris dataset [58], one of the benchmark datasets, a picture of the Eiffel Tower is given as a query and other Eiffel Tower pictures should be retrieved by the system.

With the rise in popularity of CNNs in computer vision, several works proposed to use the output of the last fully-connected layer as global image descriptors, and showed a performance improvement [26, 6, 70]. However, research has now shifted towards the use of local image descriptors of deep convolutional layers from CNNs which are aggregated into a global image descriptor [59, 5].

Two of the easiest, best performing methods [59] are the Sum Pooled Convolutional descriptors (SPoC) [5] and Maximum Activation of Convolutions (MAC) descriptors [60]. While these methods also use local image features, just as SIFT [47], their spatial description is better suited for use with similarity metrics. Babenko and Lempitsky [5] show that the individual similarities of SIFT features can be very high for very unrelated images patches [5]. On a similar note, they found that in their experiments, deeper convolutional features produce more reliable similarities than more shallow features. In their work, Babenko and Lempitsky [5] generate SPoC descriptors by first summing the `conv5_3` channels of an on ImageNet [64] pre-trained VGG-16 [71] network.

They then apply a simple Gaussian weighting scheme as centering prior as this improves the retrieval performance for most datasets. This is an artifact of the fact that most retrieval datasets have the object of interest close to the geometrical center. Finally, some post processing steps are done to obtain the image descriptors. First l2-normalization is applied, then Principal Component Analysis (PCA) is performed and the top N (256) dimensions are taken which are then whitened again l2-normalized. The PCA and whitening parameters are obtained through a left-out dataset with similar image statistics. Finally, they use the scalar product of the resulting descriptors as simple matching kernel as proposed by Bo and Sminchisescu [9]. MAC descriptors are very similar to SPoC descriptors, but instead of summing the channels, they take the maximum, non-negative, feature per channel. In their work, Babenko and Lempitsky [5] compared their SPoC descriptors to MAC descriptors using the same post processing and simple matching kernel [9]. They found that SPoC descriptors, strongly outperform MAC descriptors as well as some more complicated aggregation techniques such as Fisher vectors [57] and Triangulation embeddings [38]. Moreover, they found that the latter two suffered a lot more from overfitting, partially because of their higher dimensionality [5]. In [60] a different post processing pipeline for dimensionality reduction is utilized. In particular their approach takes advantage of labeled data. However, in an active learning setting one typically does not have a lot of labeled data. Considering this, and the fact that SPoC descriptors are compact, easy to use and have a strong performance in CBIR, we use the SPoC descriptors[5] with the simple matching kernel [9] in our work.

Note that methods that score substantially better on CBIR exist [59]. However, these often use CBIR specific methods to increase CBIR performance, which aren't necessarily directly translatable to our goal of obtaining good image representations to be used in a density metric in active learning. Examples of these methods are for CBIR fine-tuned feature extractors and spatial verification (e.g. RANSAC [11]). The reader is referred to the extensive benchmark experiments performed by Radenovic et al. [59] for different CBIR pipelines including a comparison between different feature extractors, either off-the-shelf or fine-tuned for retrieval and to the work by Seddati, Dupont, Mahmoudi and Parian [66] for a comparison including the dimensionalities of the descriptors.

3.2 Active Learning for Object Detection

Overall active learning for object detection is not a widely studied research topic. Most research only focuses on the classification uncertainty. Only a few works consider a localization component. None of these methods utilize the expressiveness of sampling-based object detectors [42, 20], nor do they combine the uncertainty with

density sampling.

In the works of Wang et al. [80, 81] *object level samples* are used. To alleviate the problems described in subsection 2.4.2, they use *pseudo-labels*, which are annotations given by the model instead of a human, for the most certain objects. Moreover, they incorporate a self-paced learning scheme, where the aim is to train on the easiest samples first and the difficulty of the objects is gradually increased. In one work, Wang et al. base the distinction between certain and uncertain predictions just on classification uncertainty [80]. In the other work, Wang et al. [81] also incorporate a localization part which they call *cross image validation*. In cross image validation, the detected object $\boldsymbol{D}_i$ is randomly pasted onto N different images. These images do not contain the predicted object class to ensure a different context. If a detection $\boldsymbol{D}_j$ on such a new image has an IoU higher than some threshold γ with $\boldsymbol{D}_i$ they calculate a consistency value ϕ_j. This consistency value denotes the possibility of $\boldsymbol{D}_j$ being of the same predicted object category as $\boldsymbol{D}_i$. This is done to find high-consistency and low-consistency predictions which are to be pseudo-labeled and actively-labeled respectively.

Another important consideration is the work of Lee, Sen and Liu [39]. They propose an active learning framework that also considers the localization uncertainty of objects. They compare random sample selection, class-uncertain sample selection and a combination of class-uncertain and two novel localization uncertainty metrics. The first localization uncertainty, *localization tightness*, measures how tight a bounding box can enclose objects of interests. It relies on the availability of ground truth bounding boxes or the use of an object detector with a separate region proposal network, such as Faster R-CNN [23]. Specifically, the metric calculates the intersection over union (IoU) of two bounding boxes B_1 and B_2 where for localization tightness B_1 is the predicted bounding box and B_2 is either the ground truth or the intermediate bounding box from the region proposal network. The other metric, *localization stability*, measures whether the model is stable to noise. First the original image, without noise, is used as input. The predicted bounding boxes that result from this are called reference boxes. Then for n noise levels, they add Gaussian noise to each pixel, with the standard deviation being proportional to n. The localization stability of each reference box is then calculated as follows:

$$S_B(B_0^j) = \frac{\sum_{n=1}^{N} IoU(B_0^j, C_n(B_0^j))}{N}$$

where B_0^j is the j-th reference box and $C_n(B_0^j)$ is the corresponding bounding box with noise level n. To combine the reference boxes to a localization uncertainty for an image i, they take the weighted average of the reference boxes, weighted by their highest class probability:

$$LS_I(I_i) = \frac{\sum_{j=1}^{M} P_{max}(B_0^j) S_B(B_0^j)}{\sum_{j=1}^{M} P_{max}(B_0^j)}$$

This ensures that images are picked with boxes that have, on average, a high probability of being foreground objects and at the same time have a high localization uncertainty. They then combine the localization uncertainty by adding adding a classification uncertainty: $LS + C = U_C(I_i) - \lambda LS_I(I_i)$, where the class uncertainty $U_C = 1 - P_{max}$ for the object on the image that has the highest class uncertainty and the lambda is set to 1. For all experiments they label the image with the most uncertain object. Similar to this work they used the SSD detector on Pascal VOC 2007, but also the Faster R-CNN [23] on MS-COCO[45] and PASCAL VOC 2007 [17] and PASCAL VOC 2012 in their experiments.

Older uncertainty-based works, prior to the rise in popularity of deep learning-based object detectors, typically used classification uncertainties obtained by sampling close to the SVM classification boundary [73, 79].

3.2.1 Probabilistic Object Detections

Probabilistic object detections are object detections that contain both a categorical distribution over the classes and a multivariate Gaussian distribution over the two object corners. To obtain probabilistic object detections, most earlier works used sampling-based object detectors to obtain a large set of detections [51, 52, 50]. For classification and segmentation it is easy to average the samples from sampling-based uncertainty techniques. However, for object detection the samples must first be correctly associated, i.e. which detections correspond to the same object, which is often done by clustering techniques [52, 51]. Then they can be merged into probabilistic object detections, also referred to as *observations* $\mathcal{O}_i$ [50]. There is however, no academic consensus yet on what the best strategy is for this procedure. This is due to the fact that the notion of probabilistic object detection is relatively new and because different strategies might perform better for different purposes.

In the work by Miller et al. [50], MC dropout with an SSD detector is used to experiment with different merging strategies. They do this on a mix of datasets, mimicking *near open-set conditions* (which include semantically similar unknown classes), *distant open-set* conditions (which include semantically dissimilar unknown classes) and *closed-set conditions* (where testing is in distribution with the training environment), in order to test the quality of the uncertainties obtained [50].

They experiment with four different clustering methods and also combine these with 2 different semantic affinity measures. The clustering techniques they examine are the Basic Sequential Algorithmic Scheme (BSAS) [51], the Hungarian algorithm [41], and the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [12] algorithm. The semantic affinity measures are Kullback-Leibler divergence between the class distributions and the 'same label' affinity, which is just a hard criterion that the highest probability classes for all detections used to obtain $\mathcal{O}_i$ must be the same. They use uncertainty error (UE) to evaluate the quality of the probabilistic object detections:

$$UE(\sigma) = 0.5\frac{|\mathcal{D}_c > \sigma|}{|\mathcal{D}_c|} + 0.5\frac{|\mathcal{D}_i \leq \sigma|}{|\mathcal{D}_i|} \tag{3.1}$$

where σ is an uncertainty threshold, the first summand is the proportion of the correct detections that are incorrectly rejected and the second summand is the proportion of the incorrect detections that are incorrectly accepted. They found the Basic Sequential Algorithmic Scheme (BSAS), with IoU (of 0.95) and 'same label affinity measures performed best overall.

In a follow-up paper, Miller et al. [52] research different sampling-based object detectors, MC dropout (40 passes) or Deep ensembles (ensemble of 5) for both the SSD and Faster R-CNN object detection frameworks. They use two different merging strategies. Firstly, BSAS clustering with 'same label' affinity using an IoU value of 0.5 as they empirically determine this performed well for every probabilistic object detector. Secondly, they propose a new merging strategy, *pre-NMS averaging*, which can only be used by one-stage detectors. In pre-NMS averaging, before detections are run trough NMS, which would result in a varying number of object proposals which then need to be correctly associated, the class distributions of the detections are averaged. Compared to BSAS clustering, its advantages are that it is faster, because there is no need for the association of different proposals and that it only requires one hyperparameter (minimum non-background softmax score for a detection), instead of 3 (minimum IoU threshold, minimum number of detections per probabilistic detection and minimum non-background softmax score for a detection) for the BSAS algorithm. The main evaluation metric is Probabilistic Detection Quality (PDQ). PDQ measures not only whether the detector accurately classifies all known objects, but also rewards an accurate expression of spatial and label uncertainty for correct detections [27]. For more information regarding PDQ, the reader is referred to the work by Hall et al. [27]. They found that pre-NMS averaging has a competitive performance in terms of PDQ. A disadvantage is that the spatial

uncertainty is constrained, as the detection can only come from a single anchor box. Regarding the sampling-based uncertainty techniques, they found that the deep ensemble of 5 networks always outperformed the MC dropout network for both merging strategies both with PDQ as well as mAP.

Another way one can obtain a probabilistic object detection, is by directly predicting the multivariate Gaussian distributions that are needed for the top left and lower right corner of the bounding box, alongside the class probabilities. This circumvents the need for sampling-based object detectors and the following merging of bounding boxes. A difficulty is that one does not have probabilistic truth values. He et al. [31] solves this by proposing a new bounding box parameterization, including a variance per bounding box coordinate, following from a newly proposed bounding box regression loss. The new regression loss is a KL-divergence loss between the distribution over each individual coordinate and the ground truth bounding box coordinate, where the latter is modeled as a Dirac delta function. These changes led to an increase of 2.8% AP compared to the baseline Faster R-CNN. This method has not been tried with other object detection frameworks. Also, the quality of the spatial probabilities have not been tested yet using probabilistic object detection metrics such as PDQ.

Chapter 4

Method

In this section we will discuss our approach towards answering our research question: *Can localization uncertainty obtained with probabilistic object detection methods be used to improve active learning for object detection?* We start by discussing the object detection architecture that is used. We then discuss our proposed sample selection strategies. This is followed by our active learning pipeline and finally the experiments.

4.1 Object Detection Architecture

In this research, all experiments are performed using the Single Shot MultiBox Detector (SSD) [46] as object detectors[1]. We follow the paper exactly in terms of implementation and only alter the training regime a bit, which is elaborated on in subsection 4.4.1. The SSD is among the best performing object detection architectures [88]. We use the standard version, called SSD300, which is fast enough to be used in real-time applications while achieving a higher mAP than Faster R-CNN [62] on MSCOCO [45], Pascal VOC 2007 [17] and 2012 [18] [46]. We believe this combination of both highly accurate and fast detections makes it particularly well suited for active learning. This combination allows it to focus on the truly difficult samples, which are not only difficult because of a bad object detection architecture, while maintaining acceptable computational cost.

4.1.1 SSD architecture

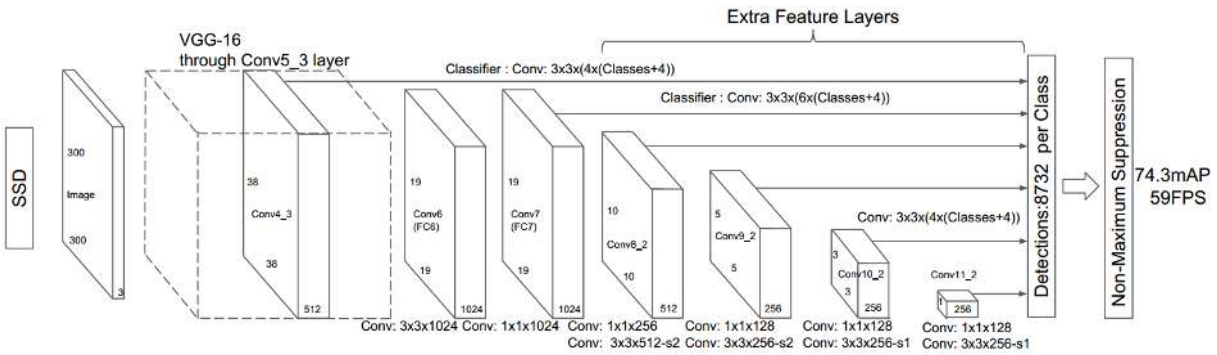


Figure 4.1: Schematic overview of the SSD object detection network as depicted in the work of Liu et al. [46]

The early layers are based on a standard architecture that is used for image classification, from here on referred to as the *base network* or *backbone*. Liu et al. [46] use a VGG-16 [71], trained on ImageNet [64] (see Figure 4.2).

[1] As a basis we use the PyTorch implementation from: `https://github.com/amdegroot/ssd.pytorch`

The fully connected layer `fc6` and `fc7` are converted into convolutional layers by subsampling parameters from `fc6` and `fc7`. `Pool5` is changed from 2×2 with a stride of 2, to 3×3 with a stride of 1. The *atrous* [33] algorithm is used to fill the 'holes'. All dropout layers and the `fc8` layer are removed. After the adjusted VGG-16 backbone, six extra convolutional layers are added which are Xavier initialized [25]. A schematic overview of the SSD can be found in Figure 4.1. Though VGG-16 is used as backbone, other networks could be used as well. This can strongly influence performance and speed [46]. Currently about 80% of time in the forward pass is spent in the VGG-16 backbone [46].

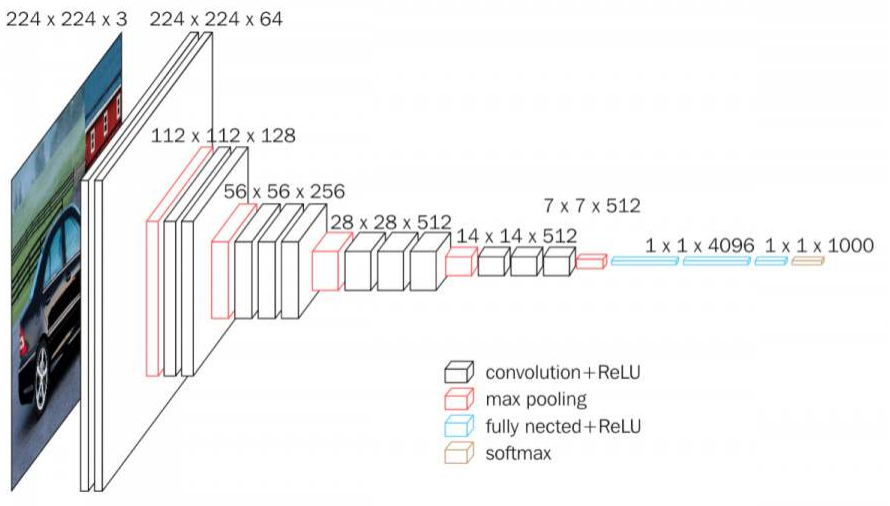


Figure 4.2: Schematic overview of the VGG-16 network

Liu et al. [46] attribute the strong performance compared to Faster R-CNN and YOLO to a series of improvements. In terms of speed, a significant speed-up is achieved by not using bounding box proposals. Though other architectures such as YOLO [61] do this as well, the elimination of bounding box proposals typically comes with a decrease in performance. Nevertheless, this drawback is being avoided by the following measures.

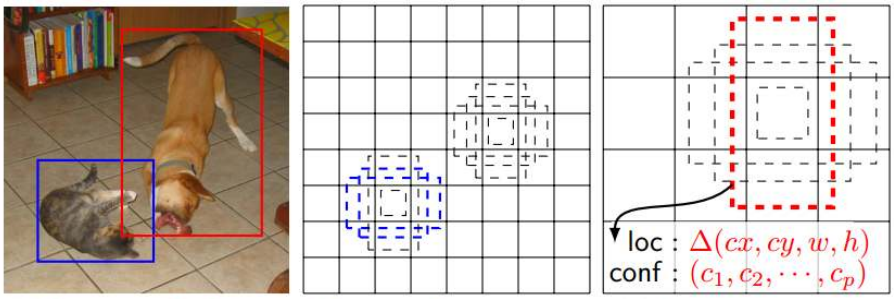


Figure 4.3: In a convolutional fashion, SSD utilizes a small set of default bounding boxes of different aspect ratios at each location in multiple feature maps of different scale (e.g. (b) 8x8 or (c) 4x4). For each default box (c), a confidence is calculated for each class, including a background class, and an offset with respect to the default box is calculated. Image taken from Liu et al. [46]

.

Firstly, *multi-scale feature maps* are used. As larger objects need larger receptive fields [46], while smaller objects are more easily detected using smaller receptive fields, the SSD uses multiple layers independently to detect features (see Figure 4.3). Previously, this was done by first processing the same image on different scales after which the results are combined [28, 67]. The multi-scale feature maps achieve the same effect in a more efficient manner. Moreover, the use of these multi-scale features allows SSD to use lower resolution images, i.e. 300x300 for SSD300 vs. $\tilde{1}000$ x 600 for Faster R-CNN and 448x448 for YOLO, increasing the detection speed even more.

Secondly, from these different feature maps, small convolutional filters are used to produce a large number of feature map cells. For each feature map cell, a set of default boxes is created. The tiling of the default boxes is designed in such a way that specific feature maps learn to be responsive to particular scales of objects. The default box shapes and sizes must be chosen manually prior to training. We chose the distribution to be exactly the same as in the work of Liu et al. [46]. This choice can be important in terms of performance and is also dependent on the dataset. For example, a different default box distribution has been shown to work better with smaller objects [30] than the one proposed by Liu et al.

The scale of the default boxes for each of the m feature maps is calculated using the following equation:

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m - 1}, \quad k \in [1, m] \tag{4.1}$$

where, s_{min} is set to 0.2 and s_{max} is set to 0.9. The lowest layer thus has a scale of 0.2 and the highest a scale of 0.9 while all layers in between are regularly spaced. Furthermore, a set of different aspect ratios are used. For most layers the aspect ratios are $a_r \in 1, 2, 3, \frac{1}{2}, \frac{1}{3}$ which can be used to compute the width $w_k^a = s_k\sqrt{a_r}$ and the height $h_k^a = \frac{s_k}{\sqrt{a_r}}$. For the aspect ratio of 1, we also add a default box with scale $s' = \sqrt{s_k s_{k+1}}$. This results in 6 default boxes per feature map cell. Exceptions are layers `conv4_3`, `conv10_2` and `conv11_2`, for which the aspect ratios $\frac{1}{3}$ and 3 are omitted, resulting in 4 default boxes per feature map. Using convolutional filters and default boxes, the image is efficiently tiled, fixing the center of each default box to the center of its corresponding feature map cell (see the second pane in Figure 4.3). At each feature map cell for each default box, bounding box offsets for bounding box regression, and class confidence indicating the presence of a class instance in a box, are calculated. By combining predictions of default boxes at different scales and aspect ratios, SSD achieves a diverse set of predictions in terms of various input object sizes and shapes. Because of the large number of boxes that are generated in SSD, 8732 for the SSD with input of 300x300, it is essential to perform non-maximum suppression (NMS) efficiently. We follow Liu et al. [46] and use a confidence threshold of 0.01 to filter out most boxes [46] and apply (greedy) NMS with an IoU threshold of 0.45 after which the top 200 detections per class are kept.

4.1.2 SSD Training

For the training procedure, we follow the work of Liu et al. [46] exactly, unless mentioned otherwise. During training, default boxes must be matched to ground truth object detections. This is done by first matching each ground truth box to the default box with the highest IoU overlap. Then all default boxes that have an overlap higher than 0.5 with a ground truth box are matched. This allows the network to predict high confidence scores for multiple overlapping default boxes, instead of having to pick only the one with the maximum overlap.

After the matching step, the majority of default boxes do not capture an object. This results in a large imbalance between positive and negative training examples. To balance this, *hard negative mining* is used. First, all negative samples are sorted by confidence loss and then only the ones with the highest confidence loss are used in the calculation of the loss. The ratio between positive and negative examples is at most 1:3.

The training objective is a weighted sum of a localization loss (loc) and a confidence loss ($conf$). Let $x_{ij}^p = \{1, 0\}$ be an indicator for matching the i-th default box with a ground truth box of class p. The loss function is:

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \lambda L1_{loc}(x, l, g)) \tag{4.2}$$

where N is the number of positively matched default boxes and λ is a weighting parameter and is set to 1. If $N = 0$, the loss is set to 0. Similar to Faster R-CNN [62], the bounding box regression is done with respect to the center (cx, cy) and the width w and height h of the default boxes d, where we only sum over the positively matched bounding boxes:

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in cx,cy,w,h} x_{ij}^{p} L1_{\text{smooth}}(l_i^m - \hat{g}_j^m) \tag{4.3}$$

where:

$$\hat{g}_j^{cx} = \frac{(g_j^{cx} - d_i^{cx})}{d_i^w} \tag{4.4}$$

$$\hat{g}_j^{cy} = \frac{(g_j^{cy} - d_i^{cy})}{d_i^h} \tag{4.5}$$

$$\hat{g}_j^{w} = \log\left(\frac{g_j^w}{d_i^w}\right) \tag{4.6}$$

$$\hat{g}_j^{h} = \log\left(\frac{g_j^h}{d_i^h}\right) \tag{4.7}$$

The localization loss is a smooth L1 loss between a predicted box l and the matched ground truth box g for every coordinate:

$$L1_{\text{smooth}}(d) = \begin{cases} \frac{1}{2}d^2 & \text{if} |d| \leq 1 \\ |d| - 0.5 & \text{otherwise} \end{cases}$$

The confidence loss is a cross-entropy loss over multiple class confidences (c):

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} \log(\hat{c}_i^p) - \sum_{i \in Neg} \log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp^(c_i^p)}{\sum_p \exp(q_i^p)} \tag{4.8}$$

where c is a

Finally, data augmentation is used during training. This makes the model more robust to various input sizes and object shapes and improves the performance substantially, taking the Pascal VOC 2007 mAP from 65.5 to 74.3 [46]. For each image, one of the following 'zoom-in' options are randomly chosen:

1. Use the original input image.
2. Sample a patch of the image, such that the IoU with at least one ground truth object is either: 0.1, 0.3, 0.5, 0.7 or 0.9.
3. Randomly sample a patch.

For each sampled patch, the size is in the range of 0.1 to 1.0 of the original image and the aspect ratio between $\frac{1}{2}$ and 2. After sampling a patch, the patch is resized to a fixed size, e.g. 300x300, and is horizontally flipped with a probability of 0.5. Finally, some photo-metric distortions are applied, following [35]. The distortions are a change in the contrast, lighting, saturation, brightness and color. The order of distortions is chosen randomly

and the magnitude of each distortion is randomly chosen between 0.5 and 1.5 where 1.0 means no distortion [35]. Aside from these 'zoom-in' data augmentation options, a 'zoom-out' option is also deployed. Zooming out is done with a probability of 0.5 by placing the image on a canvas of 16x times the image filled with mean values. This is done prior to any zoom-in operation. This creates more smaller training examples resulting in a further increase of about 2-3% mAP across different datasets. This data augmentation is a similar to what YOLO uses, but more extensive compared to Faster R-CNN, making earlier comparisons in terms of reported mAP to the latter a bit skewed.

4.2 Obtaining Probabilistic Object Detections

In this section we discuss two methods we attempted to obtain probabilistic object detections. Firstly, by utilizing various merging strategies for detections obtained with a deep ensemble of SSDs and secondly by using probabilistic bounding box regression.

4.2.1 Deep Ensemble of SSDs

In order to obtain probabilistic bounding boxes, sampling-based object detectors are often used [50, 52, 27] such as a Deep Ensemble [42] or an MC-Dropout [21] network. They are used to obtain a large set of different detections which must first be properly associated with each other, meaning that we want to obtain sets of detections $\boldsymbol{D}_i = \{\boldsymbol{c}_i, \boldsymbol{b}_i\}$ that consider the same object, where $\boldsymbol{D}_i$ is a single detection and $\boldsymbol{c}_i$ is the probability distribution for the default box and $\boldsymbol{b}_i$ are the predicted bounding box coordinates. Let us denote a set of associated detections as $\mathcal{C}^i = \{\boldsymbol{D}_1, ..., \boldsymbol{D}_N\}$ where N is the number of associated detections. From the sets of associated bounding boxes probabilistic object detections, also referred to as observations $\mathcal{O} = \{\mathcal{B}, \boldsymbol{q}\}$ with $\mathcal{B}$ as the probabilistic bounding box and $\boldsymbol{q}$ as the class distribution [52, 52, 51], can be made. In probabilistic object detection, the bounding boxes are formalized as $\mathcal{B} = (\mathcal{N}_1, \mathcal{N}_2) = (\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0), \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1))$, s.t. $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the mean and covariances for a multivariate Gaussian distribution describing the top-left and bottom-right corner of the box respectively.

This process of obtaining probabilistic bounding boxes through a large set of objects obtained by sampling-based object detectors is referred to as *merging* [52, 50]. In this work we use a deep ensemble of SSDs as Miller et al. [52] found them to outperform the MC-Dropout variant in terms of high quality probabilistic bounding boxes and have shown to outperform MC-dropout for active learning in image classification [8]. In their experiments, Miller et al. use an ensemble of 5 SSDs. In our work we use an ensemble of 6 SSD detectors in all experiments. We considered three of the most promising merging methods proposed by Miller et al. [50, 52, 51]. Though some of the best performing merging strategies also used semantic affinity measures [50, 52], we chose to not consider these. Our goal is not to obtain the best merged observations, but observations that express the best localization uncertainty. If we would include semantic affinity measures, then two spatially similar detections might not be merged. Even though the object in these bounding boxes might be spatially difficult. Furthermore, as explained in section 4.3, we can also combine the localization uncertainty with a classification uncertainty afterwards.

The BSAS merging method is a simple sequential clustering algorithm. For each detection, if an affinity measure between the detection $\boldsymbol{D}_i$ and all detections in an existing cluster C_j meets a threshold θ, then the detections is added to the cluster, otherwise, the detection is allocated to a new cluster. It can be summarized as follows [51]:

$$\mathcal{C}_i = \bigcup \boldsymbol{D}_i \quad s.t. \quad IoU(\boldsymbol{D}_j, \boldsymbol{D}_k) \geq \theta \quad \forall \boldsymbol{D}_j, \boldsymbol{D}_k \in \mathcal{C}_i \tag{4.9}$$

where θ is the IoU threshold for a detection to belong to a cluster of associated bounding boxes $\mathcal{C}$. Setting θ is very important. A higher θ leads to a higher number of observations as more clusters are generated. This results in possibly multiple observations per object. Vice versa, a lower θ could lead to a single observation for more than one object, especially when objects on the image are grouped closely to each other. Following Miller et al. [52], we set the θ to 0.5, which they found to be the best performing value and is the IoU threshold used for our evaluation metric. A disadvantage of this method is that it is potentially very computationally expensive. In the worst case, for each detection in $\boldsymbol{D}$, the IoU must be calculated for each detection in a cluster $\mathcal{C}$ for all clusters. For our active learning setting, this is amplified by the fact that especially in early stages of active learning, the SSD has many predictions that are not filtered away by the confidence threshold. Another disadvantage of BSAS clustering is the fact that a decision on the cluster-membership for $\boldsymbol{D}_i$ is made prior to the final cluster formation, which might lead to sub-optimal clustering. After clustering, we delete all clusters with less than 2 detections, as we cannot convert those into probabilistic object detections.

The second merging method we use is the HDBSCAN [13] clustering algorithm. It is a density-based clustering algorithm that allows for clusters of varying density. It does not require a pre-defined number of clusters and is robust to noisy data by leaving out outlier data points, which in our case could be bad detections. HDBSCAN calculates its own affinity measure between points, however, it is restricted to the use of two-dimensional data points. Therefore, Miller et al. experimented with [50] three, two-dimensional spatial representations of bounding boxes: the bounding box centroids, top-left corners and euclidean distance between corners and the image boundary. Of the three spatial representations, using top-left corners worked best in terms of mean Average Precision and Uncertainty Error [50]. Therefore, we also use the top-left corners as input to the HDBSCAN algorithm. As HDBSCAN does not have an IoU threshold, it can cluster objects that are very far apart from each other. This allows it to have numerous observations that, compared to the ground truth detections, have a high IoU while they have a very high spatial variance [50]. This in contrast to BSAS clustering, which does not have spatially uncertain clusters for observations with a high IoU with the ground truth object [50], while it does have spatially uncertain clusters for objects that have a low IoU with the ground truth object. This is an important observation, as for our localization uncertainty metric (see section 4.3), we utilize the spatial uncertainty present in the observations. Thus for observations that have a high IoU with the ground truth, we would want a low spatial uncertainty as these are apparently easier objects for the object detectors which we thus do not want to select for annotation. We use the implementation by McInnes, Healy and Astels [49] and keep the standard parameters except for the minimum cluster size, which is set at 2 for the same reason as with BSAS clustering.

The final merging strategy we consider is *pre-NMS averaging*, which can only be used by sampling-based one-stage detectors. One-stage detectors such as SSD tile the image using a fixed, predefined set of default boxes $\{\boldsymbol{b}_1, ..., \boldsymbol{b}_i, ..., \boldsymbol{b}_{max}\}$. Before detections are run trough NMS, which would result in a varying number of object proposals, all default boxes with the same index i are merged. Compared to BSAS clustering, its advantages are that it is faster, because there is no need for the association of different proposals and that it only requires one hyperparameter (minimum non-background softmax score for a detection), instead of 3 (minimum IoU threshold, minimum number of detections per probabilistic detection and minimum non-background softmax score for a detection) for the BSAS algorithm. Compared to HDBSCAN, it does not associate detections that are far away from each other. A disadvantage is that the spatial uncertainty is constrained, as the detection can only come from a single default box location, scale and aspect ratio. Nevertheless, Miller et al. [52] found that pre-NMS averaging has a competitive performance in terms of PDQ, which indicates that the quality of

the spatial uncertainty should be on par with that of a well hyperparameterized BSAS algorithm.

Finally, each set of associated bounding boxes $\mathcal{C}_i$ is transformed into probabilistic object detections $\mathcal{O}_i$ as follows. For each $\boldsymbol{D}$ in $\mathcal{C}_i$ let the upper left corner be denoted as a (x, y)-coordinate vector $\boldsymbol{v}_0 = [x_0, y_0]$, where the 0 indicates that it is a coordinate for the upper left corner. We then have:

$$\boldsymbol{\mu}_0^i = \frac{1}{N} \sum_{j=1}^{N} \boldsymbol{v}_0^j \tag{4.10}$$

$$\boldsymbol{\Sigma}_0^i = \frac{1}{N} \sum_{j=1}^{N} \boldsymbol{v}_0^j \boldsymbol{v}_0^{j^T} - \boldsymbol{\mu}_0 \boldsymbol{\mu}_0^T \tag{4.11}$$

The numbers on the diagonal of the covariance matrix then indicate the variance for the top-left corner in the x and y direction. Similarly the mean and covariance of the bottom right corner can be obtained by using their respective (x, y)-coordinates instead. Then for the class distribution, the vector of class probabilities $\boldsymbol{q}^i$ is obtained by averaging the probability distributions of the associated object detections [74, 51].

4.2.2 SSD with Probabilistic Bounding Box Regression

A second way of obtaining probabilistic object detections is to directly predict the probabilistic bounding boxes. We did this by adapting the work of He et al. [31] from Faster R-CNN to the SSD detector. An advantage of obtaining probabilistic bounding boxes in this manner is that you do not need, computationally costly, sampling-based object detectors. We adapted their method by firstly reformulating the bounding boxes coordinates as (x_1, y_1, x_2, y_2). Besides the coordinates, a variance σ was predicted for each coordinate. During training, the smooth L1 localization loss was replaced by their Kullback-Leibler (KL) divergence bounding box regression loss. This models the loss as the KL divergence between the bounding box distributions and the ground truth bounding box coordinates, which are modelled as a Dirac delta function. For each single coordinate x of a single predicted bounding box this is:

$$L_{reg} \propto \frac{e^{-\alpha}}{2}(x_g, x_e) + \frac{1}{2}\alpha \tag{4.12}$$

and if $|x_g - x_e| > 1$, a similar loss to the smooth L1 loss is used:

$$L_{reg} = e^{-\alpha}(|x_g - x_e| - \frac{1}{2}) + \frac{1}{2} \tag{4.13}$$

where x_g and x_e are ground truth and predicted coordinate offsets compared to the default box and $\alpha = log(\sigma)^2$ which is predicted during training to avoid gradient exploding. The α is converted back during testing and initialized with random Gaussian initialization with a standard deviation of 0.0001 and a mean of 0. Other than that, the network and training procedure was kept the same as mentioned in subsection 4.1.2

Unfortunately, in my experiments the performance of the SSD using their probabilistic bounding box regression was about 20 points lower in mAP than when using the normal smooth L1 localization loss [46]. This might be because of a combination of the following two differences of my work and theirs. Firstly, the large difference in the number of boxes on which bounding box regression is applied in Faster R-CNN compared to SSD.

Secondly, the fact that we did not give a large amount of samples to train on, because we wanted to use it in an active learning setting. Because of these reasons it might be that not all regression parameters were trained properly. Because of the large difference in mAP, we decided to not further use this method for active learning experiments.

4.3 Sample Selection Methods

In this section we discuss the sample selection methods used in our experiments. First we discuss our uncertainty metrics and how they can be combined then we discuss our density metric and how it can be combined with the uncertainty metrics.

4.3.1 Uncertainty Metrics

Inspired by the work on probabilistic object detection [52, 27, 74, 51], we define a new localization uncertainty metric to be used for object detection. As explained in section 2.3, probabilistic object detection involves predicting a probability distribution over the classes and over the bounding boxes. The classes are categorically distributed and the bounding boxes are described by two multivariate Gaussian distributions, one for the upper left corner and one for the lower right corner of the box. The traces of these covariance matrices indicate the variances in the horizontal and vertical direction for both corners. In our work, the sum of these traces thus indicate to what extent the location differs according to the SSDs.

The newly proposed localization uncertainty for a probabilistic object detection $\mathcal{O}_j$ is the following:

$$U_{loc}^{obj}(\mathcal{O}_j) = (1 - p_{bg}(\mathcal{O}_j)) \cdot \left(\frac{\text{trace}(\mathbf{\Sigma}_0^j) + \text{trace}(\mathbf{\Sigma}_1^j)}{\text{size}(\mathcal{O}_j)} \right) \tag{4.14}$$

where $p_{bg}(\mathcal{O}_j)$ is the predicted probability that an observation corresponds to the background class. The purpose of this factor is to assign higher weight to the localization uncertainty of observations that are probably foreground objects. As typically the vast majority of default boxes corresponds to background compared to foreground objects [46], we assume that the SSD learns to quickly determine a default box to be background reasonably well, also in early active learning iterations. Σ_0^j indicates the covariance matrix of the upper left corner and Σ_1^j indicates the covariance matrix of the lower right corner and $size(\mathcal{O}_j)$ is the euclidean distance between the means of the corners $\boldsymbol{\mu}_0^j$ and $\boldsymbol{\mu}_1^j$. We normalize by the sizes as the variance typically tends to scale with the object size. This would lead to a bias in favor of larger objects. However, this does not necessarily indicate it being a hard or informative object. In fact, typically smaller objects are more difficult for object detectors [88].

We also define a classification uncertainty for observations:

$$U_{class}^{obj}(\mathcal{O}_j) = (1 - p_{bg}(\mathcal{O}_j)) \cdot U_{fg}^{cl}(\mathcal{O}_j) \tag{4.15}$$

where, U_{fg}^{cl} indicates the classification uncertainty calculated only over the foreground classes, re-normalized after excluding the background probability. We exclude the background probability as it is already present in the $(1 - p_{bg}(\mathcal{O}_j))$ term. In all experiments we use entropy: $H(\mathcal{O}) = -\sum_{c \in Classes} p(c|\mathcal{O}) \cdot \log(p(c|\mathcal{O}))$ as

classification uncertainty, but other classification uncertainties could be used as well. As we use image level samples in this work, we need to propagate the uncertainties of the observations to an image level uncertainty. We do this by taking the weighted average for each image I:

$$U_{loc}(I) = \frac{\sum_{j=1}^{M} U_{loc}^{obj}(\mathcal{O}_j))}{\sum_{j=1}^{M}(1 - p_{bg}(\mathcal{O}_j))} \tag{4.16}$$

$$U_{class}(I) = \frac{\sum_{j=1}^{M} U_{class}^{obj}(\mathcal{O}_j))}{\sum_{j=1}^{M}(1 - p_{bg}(\mathcal{O}_j))} \tag{4.17}$$

Other methods could be thought of as well, such as selecting the uncertainty value of the most predicted object or the summed uncertainty. However, we find the weighted average a good compromise. When selecting the uncertainty value of the most uncertain prediction, you might sample images with one uncertain object and many non-uncertain objects. Moreover, it might give bad active learning performance if the merging of object detections would result in a few bad, extremely uncertain observations. Which might be mitigated in the case of averaging or summing. When summing the uncertainty of all predictions, you might end up sampling images with many objects which are all relatively certain.

Finally the localization uncertainty and classification uncertainty can be combined:

$$U(I) = U_{class}(I)_{normalized} + \alpha U_{loc}(I)_{normalized} \tag{4.18}$$

Where α is a weighting parameter, which is set to 1 in all experiments. Furthermore, both uncertainties are normalized to have zero mean and unit variance for the pool of unlabeled data.

4.3.2 Density and Uncertainty

In order to combine density and uncertainty, we use something similar to the *information density* metric [69, 43]. We choose this method as compared to clustering based methods it allows for very clear interpretation of the relative importance of the uncertainty and the density metrics. Because, even for clustering algorithms that do not require a predefined number of cluster centers, it is not trivial to combine these methods in such a way that there is a clear contribution from both the density and the uncertainty metric. For example, if one selects the most uncertain samples in each cluster, the spread over the data distribution is probably good, but there is no certainty that the samples within these clusters are representative samples within their cluster. Vice versa, if one would for example sample the cluster centers of the most uncertain clusters, those samples are not necessarily very uncertain.

We modify the *information density metric* slightly, instead of multiplying, we normalize and add the two terms in order to have more control regarding the weighting of the two parts:

$$ID(I) = U_{normalized}(I) \cdot \beta + \gamma_{normalized}(I) \cdot (1 - \beta) \tag{4.19}$$

$$\gamma(I) = \frac{1}{U} \sum_{u=1}^{U} sim(I, I^u) \tag{4.20}$$

where $\phi_{normalized}$ is an uncertainty metric and $\gamma_{normalized}$ is the density metric, both normalized to have zero mean and unit variance. U is the unlabeled data pool, β is a weighting parameter, which is set to 0.5 in all experiments, and $sim(\cdot)$ is a similarity metric. By doing experiments with only density as well as the combination of localization uncertainty and density sampling, we aim to obtain insights whether the newly proposed localization uncertainty metric is biased towards a certain type of images, perhaps even outlier images. For example, if active learning performance is higher when using the combination of density and uncertainty compared to when only using uncertainty, this is an indication that the sampling strategy might sample outliers.

In order to obtain a similarity between images, we first need to get a meaningful image description. In this work we use sum-pooled convolutional features (SPoC) image descriptors [5]. For an image I, we obtain the convolutional feature maps f, with spatial coordinates (x, y), from the `conv5_3` layer of a VGG-16 network [71] pre-trained on ImageNet [64]. This convolutional layer has $C = 512$ channels and following Babenko and Lempitsky [5], images are resized to 586×586 resulting in a spatial size of $W \times H = 37 \times 37$. The construction of the SPoC descriptor starts by sum-pooling the convolutional feature maps:

$$\psi_1(I) = \sum_{y=1}^{H} \sum_{w=1}^{X} f(x, y) \tag{4.21}$$

Babenko and Lempitsky [5] then propose to use a Gaussian centering prior, as for most retrieval datasets, the objects of interest tend to be located close to the geometrical center of an image. However, we skip this step because for active learning, highly informative objects might lie in the periphery. For the rest of the steps, we follow the authors closely [5]. To obtain the final image descriptor, a few post-processing steps are applied to $\psi_1(I)$. Firstly, $\psi_1(I)$ is l_2-normalized, after which PCA compression and whitening are performed:

$$\psi_2(I) = \frac{\psi_1(I)}{||\psi_1(I)||_2} \tag{4.22}$$

$$\psi_3(I) = \text{diag}(s_1, s_2, ..., s_n)^{-1} M_{PCA} \psi_2(I) \tag{4.23}$$

where M_{PCA} is a rectangular $N \times C$ whitened PCA-matrix, N is the number of dimensions that are kept and the s_i are the associated singular values. The PCA-matrix and whitening parameters are obtained using a held-out dataset with similar image statistics. We use a held-out dataset because in case the statistics of the training and test set differ, there is a risk that we overfit on the training set [5]. N is set to 256. Finally, the descriptor is l_2-normalized a second time to obtain the SPoC descriptor:

$$\psi_{SPoC}(I) = \frac{\psi_3(I)}{||\psi_3(I)||_2} \tag{4.24}$$

For the similarity measure we also follow Babenko and Lempitsky [5], and use a simple scalar product as matching kernel:

$$sim(I_1, I_2) = \psi_{SPoC}(I_1)^T \psi_{SPoC}(I_2) \tag{4.25}$$

4.4 Active Learning Pipeline

In this section the active learning pipeline is explained. Firstly, we describe the training procedure and then we discuss a diagram showcasing the complete pipeline for the various sample selection methods.

4.4.1 Training

Training in each active learning iteration for each network in the ensemble follows the same procedure. First, the seeds of the pseudorandom number generators are set to a unique value. This ensures that each network has a different initialization of its parameters and different order in which the dataset is shown. This stochasticity has empirically been shown to be enough for deep ensembles to produce good predictive accuracy as well as good predictive uncertainty by converging to sufficiently differing local optima [42]. Then, similarly to Gal [22] and Beluch et al. [8], reset the models after each active learning iteration to a model that only has the pre-trained VGG-16 backbone. This reset is performed such that the effects of the sample selection methods on the performance can are not confused with suboptimal training performance, as bad performance could have been due to having converged towards bad local optima [22].

We carefully selected a seed set and number of epochs to train for each category. We performed a small experiment to determine the number of epochs as well as the seed set. Specifically, for each class we performed active learning using random sampling using various number of epochs $n \in [60, 80, 100, 120, 140, 160, 180, 200, 220, 240]$. Per class, we picked the number of epochs such that it had only a small difference in AP per active learning iteration compared to the performance when using $n - 1$ number of epochs. This is done to ensure that the networks are able to converge in every active learning iteration, despite the low number of samples, whilst not wasting computational resources. The n we used differs per setup and is elaborated on in subsection 4.5.2 and subsection 4.5.3. After we chose the number of epochs, we picked the samples with which we were able to achieve at least 5% AP for that object category, as a seed set. After 50% of the epochs, a validation pass is done each epoch on a separate validation set. The validation set is a set of 10 randomly selected images for each of the object categories. We chose a relatively large number of epochs, and a high frequency of validation passes to ensure that variations in performance are less likely to originate from the training procedure ending up in a bad local optimum, but because of the subset they were trained on. The state of the network that has the lowest validation loss is used in the next active learning iteration for sample selection. As proposed by Liu [46], the learning rate starts at 10^{-4} and is reduced by a factor of 10 after about 66% and 88% of the number of training epochs. The learning rate starts at 10^{-4}. In all experiments we use a batch size of 8.

4.4.2 Sample Selection

For each sample selection strategy except random, we rank the images from 'best' to 'worst' and greedily select the batch of the k 'best' samples until the annotation budget for that iteration is depleted. Following earlier work [8, 22, 7, 39, 82, 80, 81], we use batches in our active learning process. This decreases the interactiveness of the active learning. However, it reduces the computational overhead of training the models and obtaining model outputs needed to calculate uncertainties. This is because less active learning iterations are needed to obtain an observable increase in performance. This overhead is even more apparent in the case of sampling

based uncertainty methods [8, 22] where multiple forward passes are needed to obtain the uncertainty and in the case of an ensemble even multiple models need to be trained. To the best of our knowledge, all research on active learning for object detection with an image level sample definition, uses a sampling budget defined in number of images [39, 7]. However, as some images contain a lot more objects than others, we hypothesize that the number of objects on an image can be a more appropriate budget definition. If the number of images is used as budget definition, there might be an unfair advantage for sample selection strategies that tend to select images with more objects. However, reversely one can also argue that sampling more images with less objects per image can be advantageous, as the object is then seen in more scenarios. Therefore, we experiment with both budget definitions in order to get more insights into the behaviour of the sample selection strategies. To achieve the exact number of objects in the object definition case, we greedily sample the images deemed best according to the sample selection strategy, until the next image would exceed the budget. We than pick the image that is best that still fits. We continue to do this until the budget is depleted. Note that in both budget definitions every sample, being it an object or an image, is equally 'costly' to annotate. In the diagram below Figure 4.4, the complete active learning pipeline can be found. Note that during the sample selection, the only data augmentation that is performed is a resizing to 300×300.

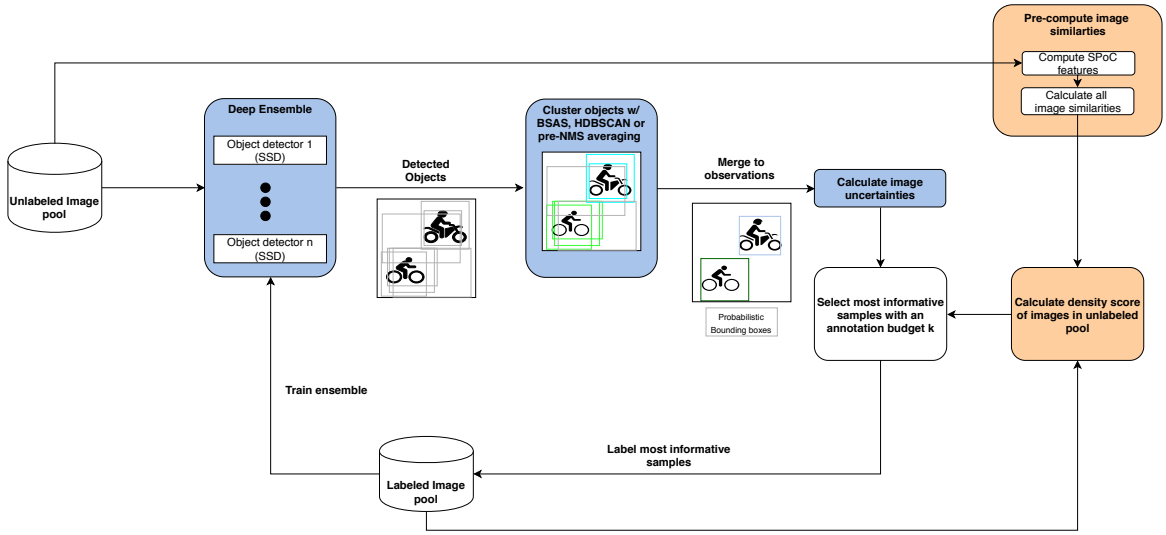


Figure 4.4: Diagram of the active learning pipeline. The blue section is only used when uncertainty is incorporated and the orange section is only used when density is incorporated.

4.5 Experimental Setup

In this section we elaborate on the experimental setup. First we discuss the dataset we use, then we explain what we base our evaluation upon and finally we discuss the baselines we compare our sampling strategies with.

4.5.1 Datasets and Image Set Splits

All experiments are done on the Pascal VOC 2007 object detection dataset [17]. The dataset is multi-class and multi-label and contains 20 object categories for which the bounding boxes are labeled. The images were retrieved from *Flickr*. For computational reasons, we conduct all experiments on a subset of six carefully selected classes. Based on the Average Precision (AP) of preliminary experiments using random sampling on all classes, we chose the two hardest, the two easiest and two random categories, respectively: *bottle, pottedplant, car , boat, sheep and horse*. An overview of the six classes can be found in table 4.1 below.

	trainv. img	trainv. obj	trainv. obj/img	seed img	seed obj	seed obj/img	val img	val obj	val obj/img	test img	test obj	test obj/img
boat	188	398	2.11	21	40	1.90	10	31	3.1	176	393	2.23
bottle	262	634	2.42	22	40	1.81	10	14	1.40	240	657	2.74
car	761	1644	2.16	15	20	1.33	10	27	2.70	775	1541	1.99
horse	294	406	1.38	9	10	1.11	10	10	1.00	279	395	1.42
pottedplant	273	625	2.29	16	30	1.86	10	13	1.30	254	592	2.33
sheep	97	353	3.64	11	30	2.72	10	24	2.40	98	311	3.17

Table 4.1: Image set statistics for the different classes. Note that not all seed sets have the same number of images and objects and that in both the *trainval* and in the *test* set there is a disbalance in the number of images and objects per class. The number of objects per image *(obj/img* columns are per class. There can be more relevant objects per image of other classes.

Note that the number of objects and images is slightly higher than mentioned on the website of Pascal VOC 2007 [1]. This is because we also include the objects that where annotated as 'difficult'. In Pascal VOC, objects are marked difficult for a number of reasons such as small size, illumination, image quality or a significant need for contextual information [17]. We chose to include these in both training and testing as we are interested to see if the uncertainty based sampling strategies are more inclined to select these objects. Moreover, in a real world setting, one would want to detect the more difficult samples as well. An example of difficult versus non-difficult annotations can be found in figure 4.5.

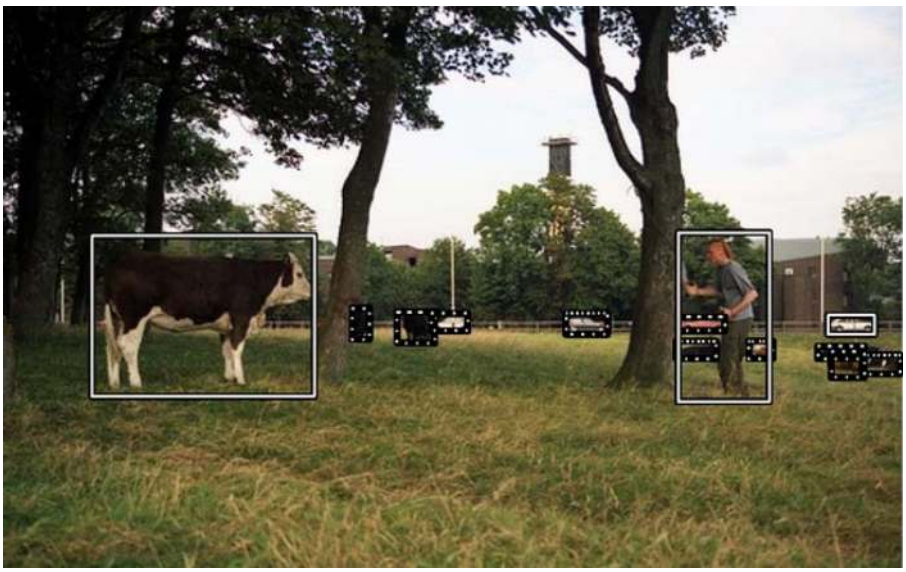


Figure 4.5: Example of difficult (dashed) and non-difficult (non-dashed) annotations in Pascal VOC. Note that difficulty is not solely based on size, as the car on the right is marked as non-difficult. Taken from [17].

We maintain the same *trainval - test* split as Pascal VOC 2007, where the images in the *trainval set* are used for active learning and the images in the *test set* are used for evaluation. To obtain the PCA-matrix and whitening parameters used to create the SPoC-descriptors, we use the Pascal VOC 2012 dataset as 'held-out dataset'. This dataset has the same classes and is obtained in the same manner as the Pascal VOC 2007 dataset, which we assume to result in similar image statistics.

We employ the *image level sample* definition described in subsection 2.4.2 and consider two different setups for the image set splits. Below we elaborate on the two setups, a multi-class and a two-class setup.

4.5.2 Two-Class Setup

Firstly, we consider a two-class setup. In this setup only a single object category is considered relevant and only the images on which they appear are considered to be in the unlabeled pool. The number of epochs n is set at $n = 200$ for *sheep* and *horse*, for all other classes we set $n = 220$. The sampling budget for all classes is set at 5 images for the image budget and 10 objects for the object budget. As some images have more than 10 objects on an image, they cannot be selected with an object budget of 10. This is a small subset, about 6%, of the images and the same for all sample selection methods. Therefore, we do not expect this to impact the results.

4.5.3 Multi-Class Setup

The second setup we consider is a multi-class setup in which all images for the 6 object categories in the Pascal VOC 2007 *trainval* are considered to be the unlabeled pool. We combine the seed sets of the two-class setup as a seed set for the multi-class setup. We checked for overlap between the combined seed set and validation set, but there is none, so they can be safely used. This setup allows for the use of classification uncertainty, as is typically done in active learning for object detection [81, 80, 39, 73, 79]. A disadvantage is that performance is harder to compare between the different sample selection strategies. If a certain sample selection strategy is sampling more easier classes, its mean average precision (mAP) might increase more strongly. However, this is not necessarily because it samples informative instances. In all multi-class experiments we trained the networks for 200 epochs for this setup we noticed in early experiments that after 200 epochs the validation loss almost never decreased anymore. Longer training was mostly a waste of computational resources, while being fairly

certain that with 200 epochs the SSDs converged as well. Due to computational limitations we only apply the image budget. We choose the image budget here as this is the standard in the research community. We use a budget of 25 images every iteration.

4.5.4 Baselines

The first baseline is a random sample selection. Note that the random baseline is harder to beat than it might seem on first eye. As every image in the unlabeled pool has relevant objects, randomly selecting images results in a good representation of the complete dataset. The second baseline is the information density metric Equation 4.19 with β set to zero, such that only density sampling is used.

For the multi-class setup, another baseline is to use only classification uncertainty as in equation Equation 4.17. A final baseline we wanted to use is the Localization Stability, proposed in [39]. It is the only localization uncertainty metric used in active learning for object detection that uses the image level sample definition. However, as we were not able to reproduce their results, we refrain from including it in the comparison. Unfortunately, we therefore cannot directly compare our result with earlier work as none use our specific image set split [81, 80, 73, 79, 39]. Moreover, if we would have had enough computational resources to also perform an experiment on the full dataset, most other papers do not use the SSD detector [81, 80, 73, 79], making comparisons based on reported results meaningless.

4.5.5 Evaluation

The evaluation metric we use is the numerically integrated Pascal VOC version of AP, as is explained in subsection 2.2.3. For all experiments a sampling strategy is considered better when it clearly outperforms the other methods in a majority of the active learning iterations. In the two-class experiments we follow earlier work [20] and perform 3 runs per sample selection strategy. Due to computational reasons, we perform 2 runs for per sample selection strategy in the multi-class experiments. The only exception is when we only use density sampling, where we perform a single run as there is no stochasticity involved in the sample selection. For all sample selection methods, 6 SSDs are trained each active learning iteration and their APs are averaged. This is done to further increase the likelihood that good performance is due to the sample selection and not because the training procedure ended up in a bad local optimum.

Chapter 5

Results

In this section we discuss the results. We start with all two-class experiments followed by the multi-class experiments. Then we perform an ablation study of our localization uncertainty metric and we conclude with a qualitative analysis.

5.1 Two-Class Setup

For the two-class experiments, we begin by comparing the different merging methods and for each merging method we research whether adding density sampling aids the active learning. Then we compare the best performing merging method with the baseline methods. As we compare many different sample selection methods, we display less cluttered plots containing subsets of the graphs in the following sections. In the Appendix, the results of all sampling methods can be found for both the object budget definition (figure 7.2) and the image budget definition (figure 7.1) in single plots.

5.1.1 Merging Strategies

We first elaborate on the performance of our localization uncertainty metric (Equation 4.16) with the different merging strategies (*HDBSCAN, pre-NMS averaging* and *BSAS*) both with and without density sampling in order to gain insight in the performance and behavior of the different merging strategies. Recall that when combining the localization and classification uncertainty with the density sampling metric, the weighting parameter β in equation Equation 4.19 is set to 0.5.

The results for the image budget can be found in figure 5.1 and the results for the object budget can be found in figure 5.2.

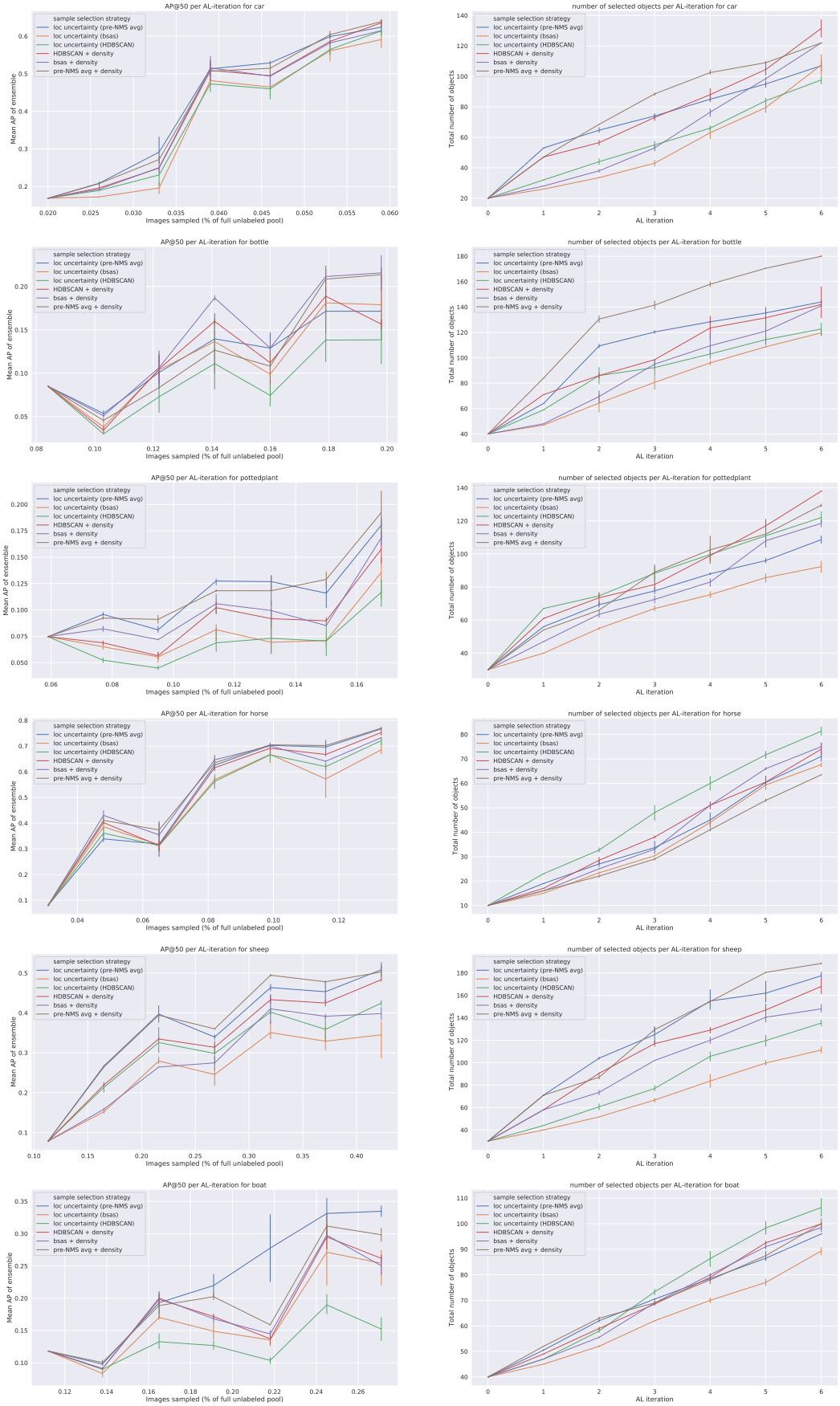


Figure 5.1: Image level budget results of the 3 merging methods with and without adding density sampling. The left column is AP averaged over the 6 SSDs and the right column shows the number of objects selected for each iteration. Each row shows the results of a different object category. The plots are interpolated between each active learning iteration and the horizontal bars are confidence intervals on the performance and number of selected objects per active learning iteration for the 3 runs per stochastic sample selection strategy. Each round 5 images are sampled. Note that the axes do not start at the origin.

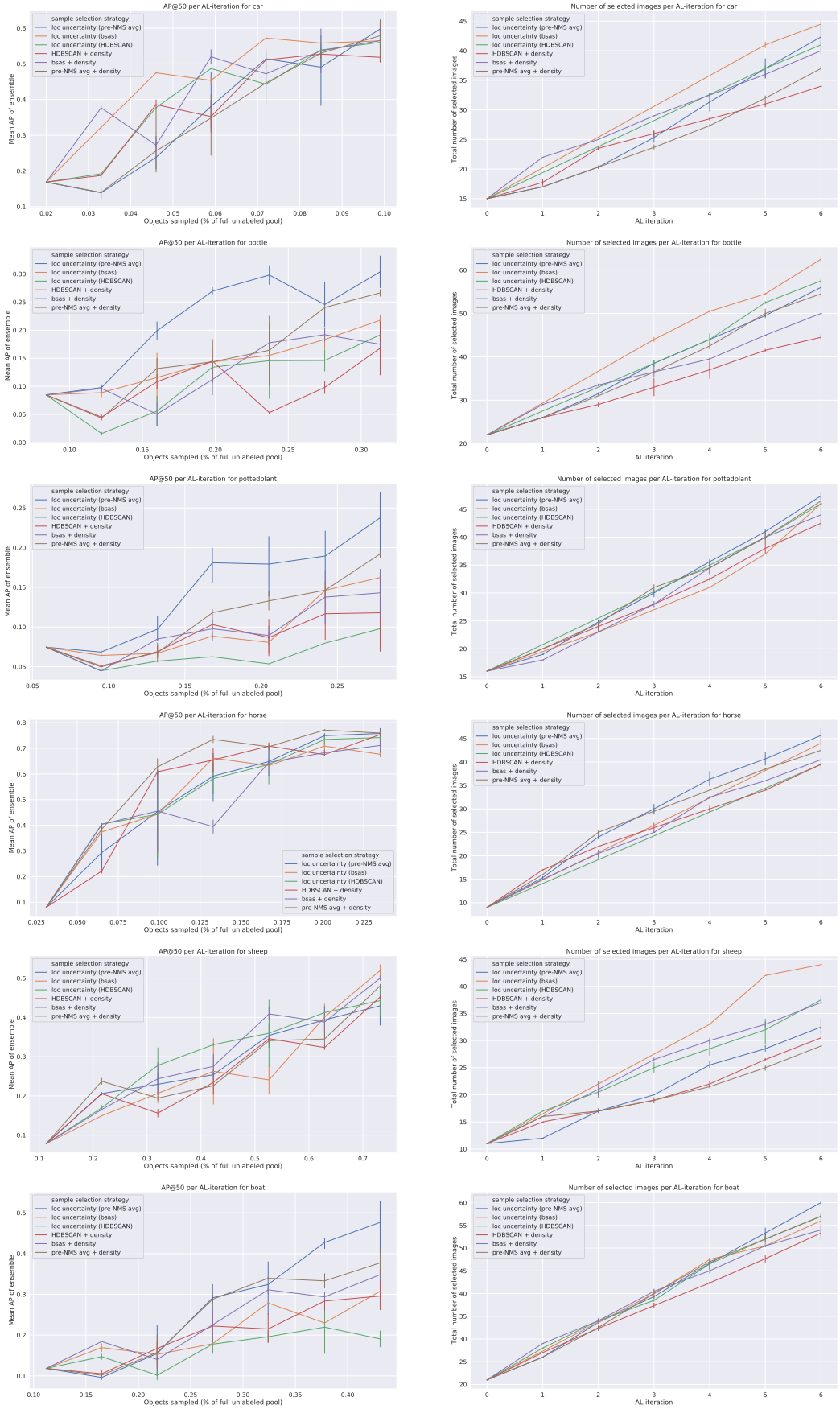


Figure 5.2: Object level budget results of the 3 merging methods with and without adding density sampling. The left column is AP averaged over the 6 SSDs and the right column shows the number of objects selected for each iteration. Each row shows the results of a different object category. The plots are interpolated between each active learning iteration and the horizontal bars are confidence intervals on the performance and number of selected images per active learning iteration for the 3 runs per stochastic sample selection strategy. Each round 10 objects are sampled. Note that the axes do not start at the origin.

Pre-NMS averaging is the best performing merging strategy in our experiments. In contrast to the other two merging strategies, the performance often drops when adding density sampling. This is an indication that it does not suffer from sampling outliers from very low density regions of the image space. Both with and without adding density sampling it almost always outperforms the other merging methods. This is also the case when density sampling is added to the localization uncertainty with the other merging strategies. Both with and without adding density sampling, *pre-NMS averaging* is only outperformed on *bottle*, by *BSAS* with density sampling. For the object budget *pre-NMS averaging* performs worse than for the image budget, but overall still outperforms the other two merging strategies. Without adding density it performs best on *boat, potted plant and bottle*. With density sampling added it performs best on *horse*. On *sheep* and *car* it is outperformed by the other merging strategies.

For the *BSAS* merging strategy, we find that it almost always samples the least amount of objects per iteration with the image budget. With the image budget it always performs better with density sampling added. This might suggest that it samples from low density areas, however, note that in our experiments density sampling tends to select images with a lot of objects on it (Figure 5.1). In the object budget experiments, we still observe that *BSAS* performs better more often when density sampling is added, though not as often as strongly as before. This suggests that the increase in performance for *BSAS* when adding density sampling is partially due to the fact that it just samples more objects in that case, but probably also because without the density sampling, it samples from lower density regions. Without adding the density sampling metric it performs better than the *HDBSCAN* merging strategy most of the times with both budget definitions, even though *HDBSCAN* always samples more objects. Compared to *pre-NMS averaging*, *BSAS* almost always performs worse. An important note on *BSAS* is that it is a lot slower than the other methods. In our experiments it took on average about 10 seconds to merge the detections for a single image, while for *HDBSCAN* it could merge the largest unlabeled dataset, that of *car*, within a second and *pre-NMS averaging* could do that almost instantaneously.

The *HDBSCAN* merging strategy is the worst performing merging strategy for almost all of the object categories with both budget definitions. This might be because *HDBSCAN* can merge detections with a low IoU, as explained in section 4.2. This results in artificially high uncertainty for probabilistic object detections, making our proposed uncertainty metric a poor indicator of the informativeness of a sample in such cases. When adding density sampling it always performs better than without when the image budget definition is used and almost always better or equal when the object budget definition is used. This is similar to the *BSAS*. However, *HDBSCAN* typically samples images with more objects on it than density sampling does. This is a strong indication that this sampling strategy tends to select images that are in a low density region of the image space. Nevertheless, when adding density sampling, the performance often is still not better than *BSAS* with density nor *pre-NMS averaging* with or without density for both budget definitions.

5.1.2 Baseline Comparison

In this section we compare the localization uncertainty (*loc uncertainty*) from the overall best performing merging method, pre-NMS averaging, with the baseline sample selection strategies: sampling at random (*random*) and density sampling (*density*). The results can be found in Figure 5.3 and in Figure 5.4 for the image budget and object budget respectively.

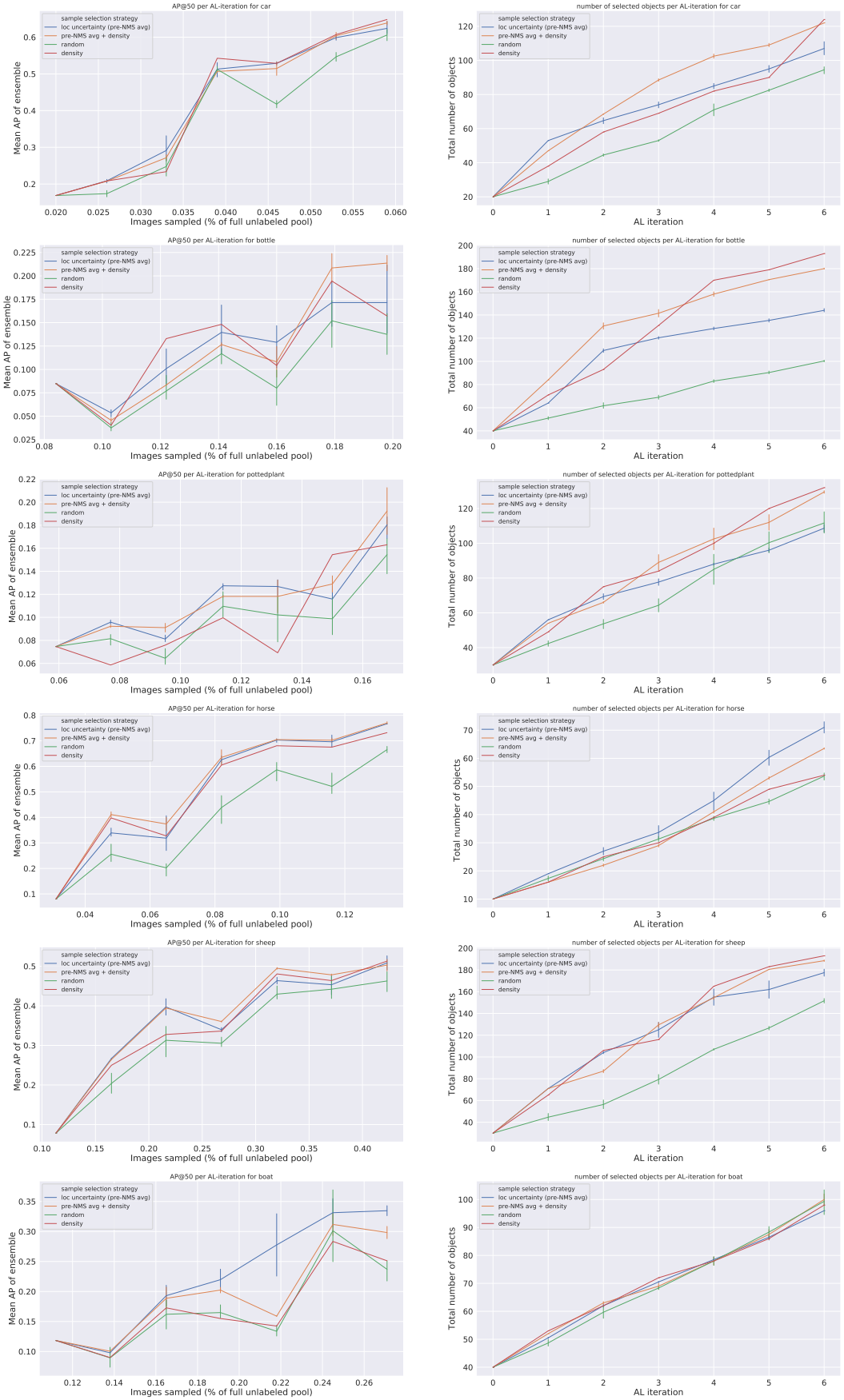


Figure 5.3: Image level budget results of the localization uncertainty with pre-NMS averaging, with and without adding density sampling compared to the random sample selection and density sampling baselines. The left column is AP averaged over the 6 SSDs and the right column shows the number of objects selected for each iteration. Each row shows the results of a different object category. The plots are interpolated between each active learning iteration and the horizontal bars are confidence intervals on the performance and number of selected objects per active learning iteration for the 3 runs per stochastic sample selection strategy. Each round 5 images are sampled. Note that the axes do not start at the origin.

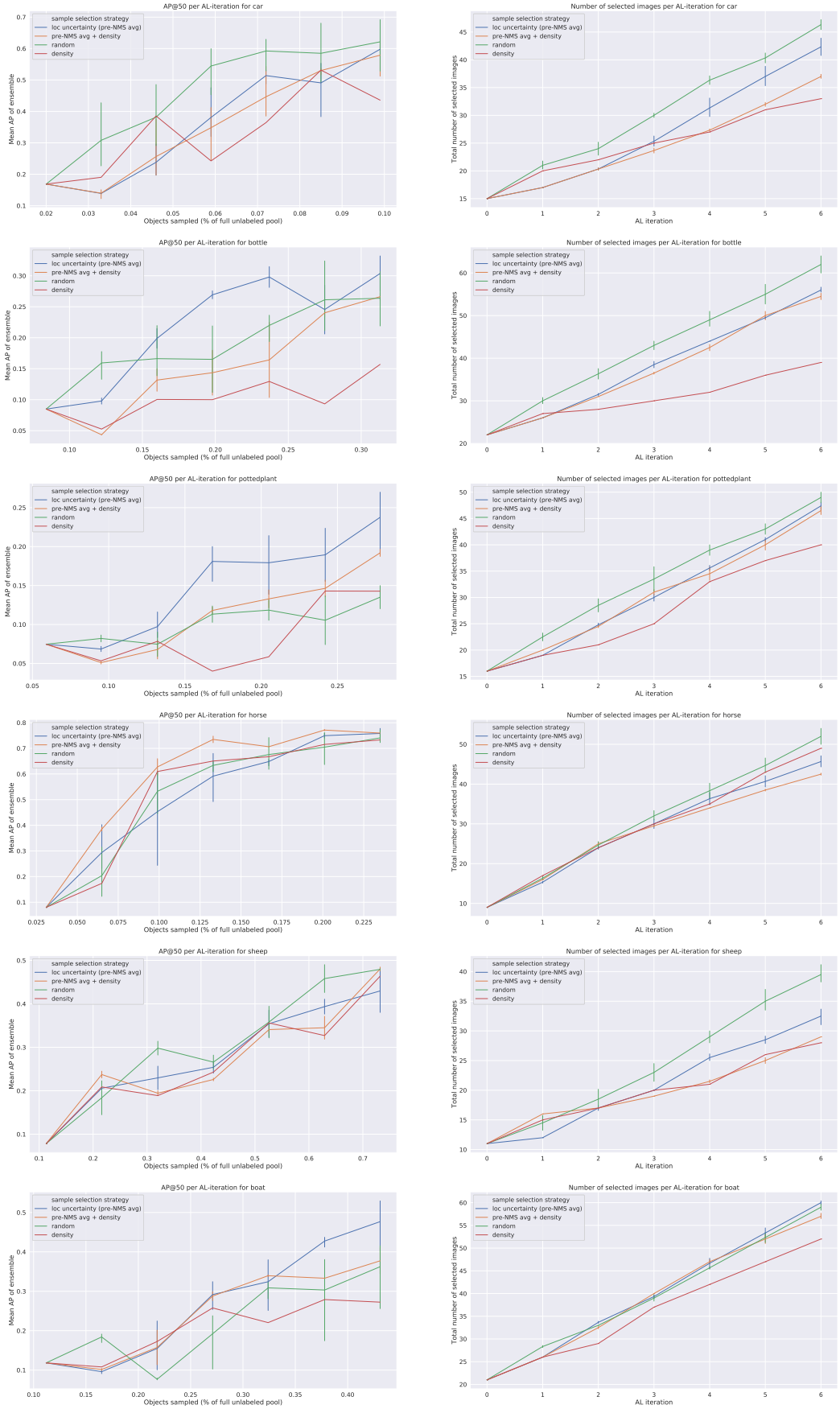


Figure 5.4: Object level budget results of the localization uncertainty with pre-NMS averaging, with and without adding density sampling compared to the random sample selection and density sampling baselines. The left column is AP averaged over the 6 SSDs and the right column shows the number of objects selected for each iteration. Each row shows the results of a different object category. The plots are interpolated between each active learning iteration and the horizontal bars are confidence intervals on the performance and number of selected images per active learning iteration for the 3 runs per stochastic sample selection strategy. Each round 10 objects are sampled. Note that the axes do not start at the origin.

In the image budget setup *loc uncertainty* performs better than *random* for all object categories, both with and without adding density sampling. In the object budget setup the performance difference is more nuanced. Our localization uncertainty metric outperforms random sampling for the two harder classes: *bottle* and *potted plant* as well as on *boat*. If density is added it also outperforms random sampling on *horse* but then it performs worse than *random* on *potted plant*. On both *sheep* and especially on *car*, *random* performs better than our localization uncertainty metric. *Density* often performs very badly when an object budget is used. This might be because it often samples images with many objects on it, resulting in very little different images to learn from and possibly overfitting on the few scenes it sees the object categories in. For the image budget, it performs relatively well compared to *random*, outperforming it in the majority of the settings. This is probably mainly because it samples more objects than *random*. However, it must be noted that on *horse*, it outperforms *random* as well, while it selects images with approximately the same number of objects.

5.2 Multi-Class Setup

In this section we discuss the results in the multi-class setup. For the localization uncertainty we only used the pre-NMS averaging method as it was the best performing method in the two-class setup. We compare the following sample selection strategies: sampling with only localization uncertainty sampling (*loc uncertainty*), with only density sampling (*density*), with only classification uncertainty sampling (*cls uncertainty*), with a combination of localization uncertainty and class uncertainty (*loc + cls uncertainty*) and with a combination of localization and classification uncertainty and density sampling (*loc + cls uncertainty + density*). Finally, we also compare these sampling methods with randomly sampling images (*random*). Recall that in all experiments where localization uncertainty is combined with classification uncertainty, the weighting parameter α in Equation 4.18 is set to 1.0. When combining any uncertainty metric, also the combined localization and classification uncertainty, with the density metric, the weighting β parameter in Equation 4.19 is set to 0.5.

The results can be found in Figure 5.5. Plots with all results of the multi-class experiments including the results from the ablation study, discussed in subsection 5.2.1, can be found in the appendix (figures 7.3 and 7.4 with or without the fully trained performance respectively).

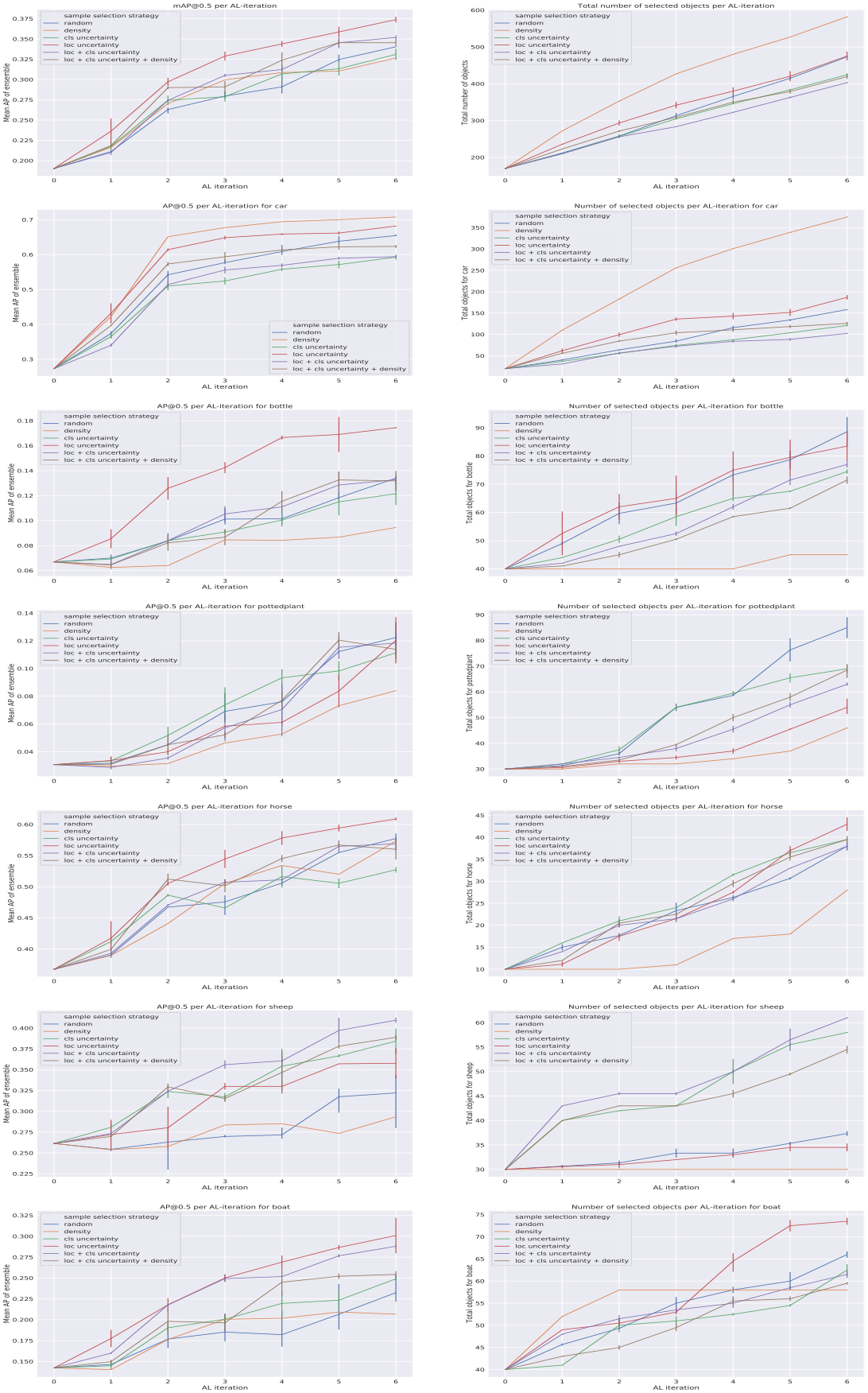


Figure 5.5: Results of the multi-class setup. The left column is the AP averaged over the 6 SSDs in the ensemble with the confidence bounds from the two runs per method. In the right column we show the number of objects that are sampled from the labeled pool for each class for each sample selection method. Each round 25 images are selected. Note that the axes do not start at the origin.

Loc uncertainty performs best in the majority of the cases, including the mAP over all 6 classes. It performs better than *random* in all cases except for *potted plant*. This is probably due to the fact that it samples very little of this class, as it does perform well on this class in the two-class setup. Moreover, it performs on par with *random* in the final iteration for *potted plant*, with just over half the number of objects sampled. It performs notably well on *bottle* where it strongly outperforms all other sample selection methods. Except for *density*, it selects images with slightly more objects per image compared to all other methods, but not a lot more. *Cls uncertainty* performs slightly better than random, outperforming it for *boat, sheep, potted plant* and on average over all 6 classes. Interestingly it performs relatively bad for *car* and *horse*, the two easiest classes in terms of AP in our experiments. It might be that the images it selects with horses and cars on it are selected because they co-occur of other objects classes it is uncertain about. In order to see whether localization uncertainty and classification uncertainty are complementary, we combine them as stated in Equation 4.18. We observe that when combining the two, it often results in a performance that is in between that of *loc uncertainty* or *cls uncertainty*, suggesting that the two are not complementary. One exception is *sheep*, in which the combination performs the best of all metrics. Recall that *sheep* is one of the harder classes for *loc uncertainty* in the two-class setup and one of the two classes where it performs worse than sampling at random in the two-class object budget setup.

The density metric we used almost exclusively samples *car*-objects and, in early iterations, *boat*-objects. For the *car* category it outperforms all other methods, but compared to *loc uncertainty*, it beats it only slightly, while using more than two times the number of objects. The reason for the bias towards *car*-objects is probably because we used a density metric over the complete image set. As can be seen in Table 4.1, *car* is the most prominent category in the dataset, resulting in an over-representation of *car*-objects when using density sampling. However, there are not a disproportionate amount of *boats* in the dataset. Probably, the images with boats are visually very similar, making them a high density region of the image space as well.

Loc + cls uncertainty + density is a sampling strategy that performs reasonably well. In terms of mAP it performs on par with *loc + cls uncertainty*. It performs relatively bad on *sheep* compared to *loc + cls uncertainty*, probably because the density sampling metric has no tendency to sample *sheep* at all. It performs better or on par with sampling using only the classification uncertainty metric. It outperforms sampling with only the localization uncertainty metric for the two object categories in which the localization uncertainty metric does not perform well: *sheep* and *potted plant*. Moreover, it always performs better or similar to random sampling, making it a low-risk sampling strategy in our experiments. Interestingly, on average over the six classes, the three sample selection strategies that include our localization uncertainty metric outperform the other strategies.

Finally, random sampling is among the worst sampling methods in most cases, especially in early iterations. The only exception where it performs reasonably well is on the *car* category. This is probably due to the fact that *car* is such a prominent object category and, when randomly sampling from the unlabeled data pool, a fair number of diverse *car* objects is sampled.

5.2.1 Ablation Study

In this section we discuss the results of our ablation study. Specifically, we compare our localization uncertainty metric (*loc uncertainty*) as in equation 4.16, with the same metric where we do not multiply by the foreground probability (*no-foreground-scale*) or where we do not divide by the size of the object (*no-size*), and with sampling at random. We perform this experiment in the multi-class setting. The results can be found in 5.6.

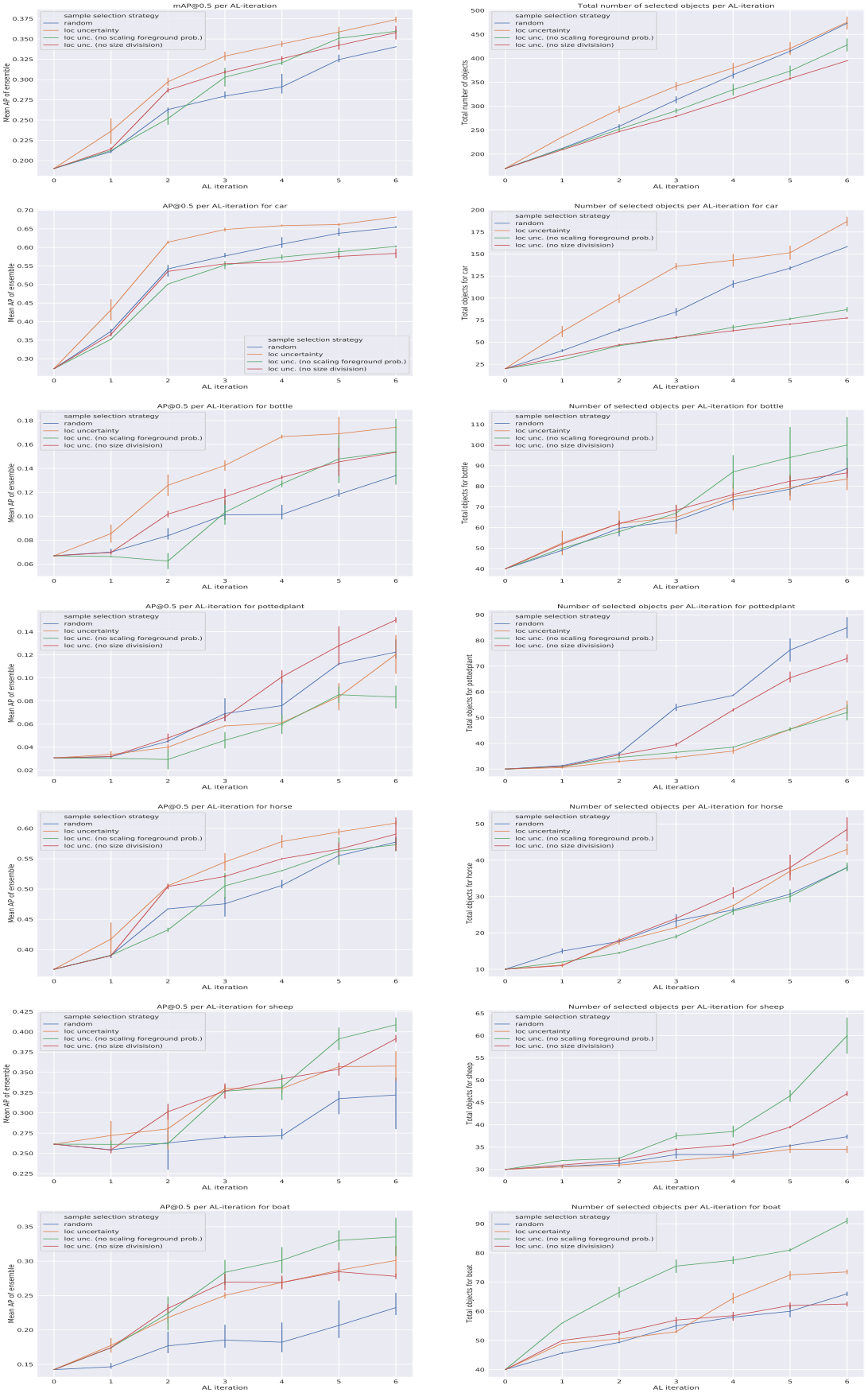


Figure 5.6: Results of the ablation study. Left column is the AP averaged over the 6 SSDs in the ensemble with the confidence bounds from the two runs per method. In the right column we show the number of objects that are sampled from the labeled pool for each class for each sample selection method. Each round 25 images are selected. Note that the axes do not start at the origin.

Interestingly, both *no-size* as well as *no-foreground-prob* perform better than sampling at random in most setups. This suggest that the localization uncertainty obtained through pre-NMS averaging with an ensemble of SSDs measures the informativeness of an observation $\mathcal{O}$ well and is useful in active learning for object detection. Nevertheless, both *no-size* as well as *no-foreground-prob* are in most cases themselves outperformed by *loc uncertainty*. In terms of number of objects sampled, *loc uncertainty* samples relatively similar amounts as *no-size* and *no-foreground-prob*, with the exception of the *car* category, in which it samples more than twice as much. Compared to *no-size* performs very similar on *boat* and *sheep* and better on *potted plant* compared to *loc uncertainty*, but is outperformed in all other cases. Compared to *no-foreground-prob*, *no-size* performs very similar with two notable exceptions. For *potted plant* it outperforms the other sampling methods in later iterations, probably because it samples relatively many of these objects. For *boat*, *no-foreground-prob* outperforms all other methods. Again this can probably be attributed to the fact that it tends to select objects of this category relatively often.

5.3 Qualitative Analysis

In this section we perform a qualitative analysis of a few interesting cases mentioned above. We do this based both on collages of images sampled by various sample selection methods and plots of the number of *difficult* and *non-difficult* objects sampled, utilizing the difficulty flag provided by the Pascal VOC dataset [17]. We start with a general discussion about the selection of difficult and non-difficult images. This is followed by two case studies where we display the selected images.

5.3.1 Difficulty of Selected Samples

In this section we discuss the distribution of difficult versus non-difficult objects per sample selection strategy. In Figure 5.7, Figure 5.8 and Figure 5.9 the results for the two-class setup with image budget, two-class setup with object budget and the multi-class setup can be found respectively. In the appendix we show the results after 6 iterations for all classes separately (Figure 7.5 Figure 7.6 , Figure 7.7,).

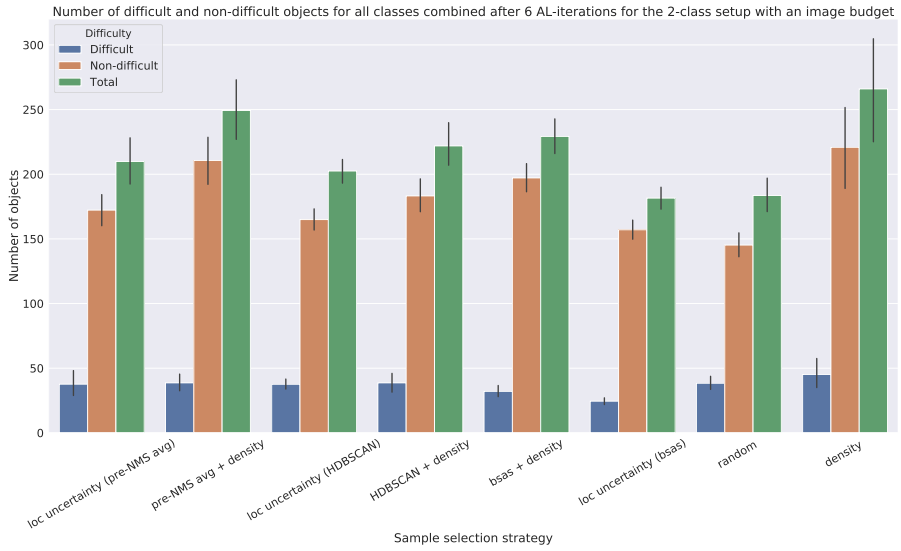


Figure 5.7: The distribution of difficult versus non-difficult sampled objects per sample selection strategy in the two-class setup with an image budget after all 6 active learning iterations. The objects from the seed-set are not included.

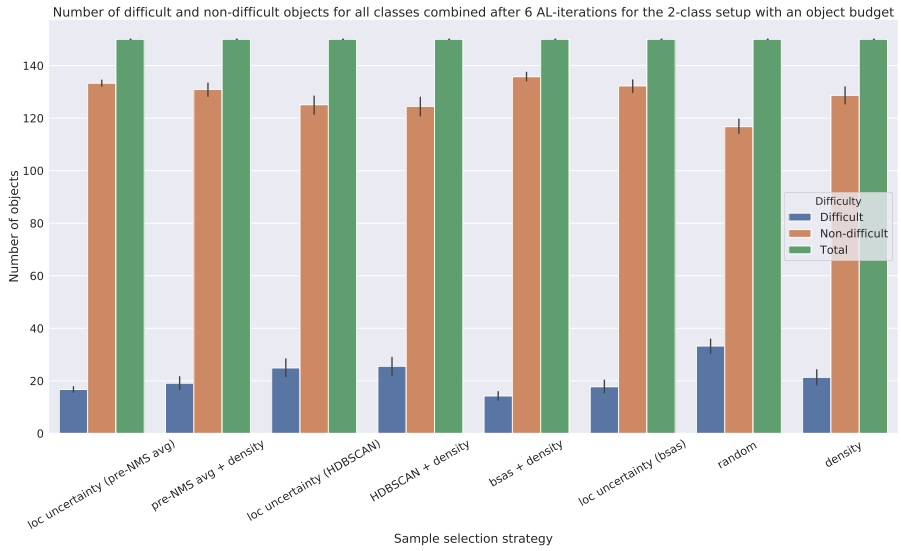


Figure 5.8: The distribution of difficult versus non-difficult sampled objects per sample selection strategy in the two-class setup with an image budget after all 6 active learning iterations. The objects from the seed-set are not included.

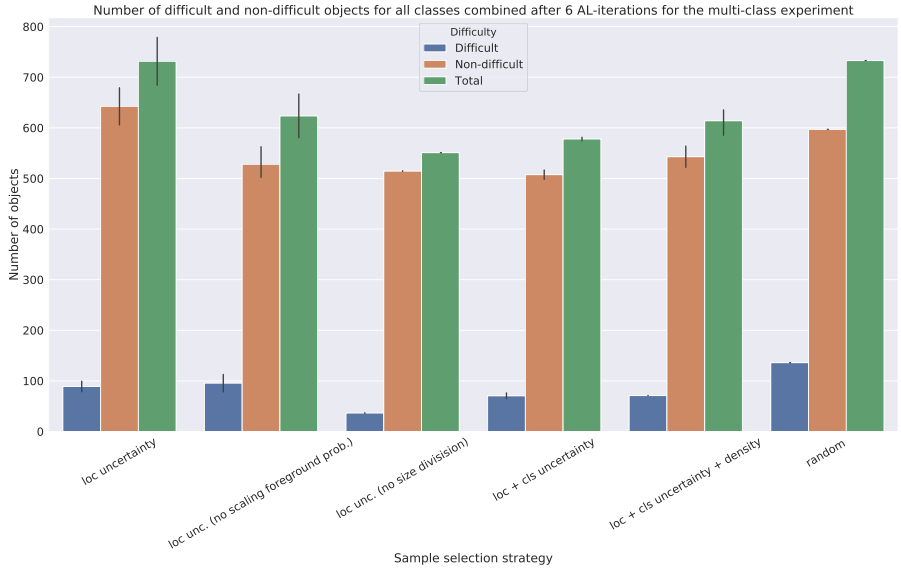


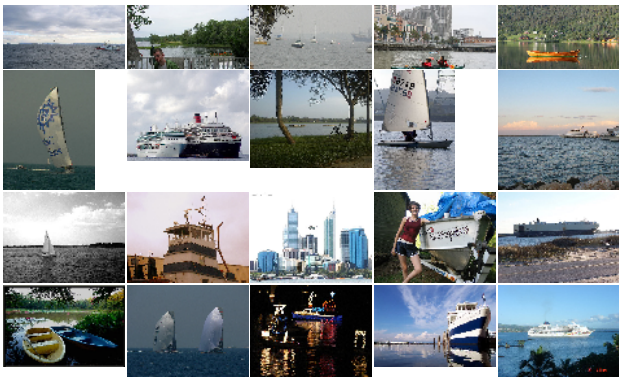
Figure 5.9: The distribution of difficult versus non-difficult sampled objects per sample selection strategy in the multi-class setup with an image budget after all 6 active learning iterations. The objects from the seed-set are not included.

For the two-class setup with the image budget, we see that the amount of difficult samples selected is fairly similar. As we did observe a notable difference in performance overall, this suggests that difficult objects do not necessarily contain the most information. In the two-class setup with the object budget, we observe that randomly selecting the samples actually always results in a higher number of 'difficult' samples. This indicates that, though it should sample objects of which the location is hard, our localization uncertainty metric does not necessarily select objects that are difficult according to the Pascal VOC definition. In the multi-class setup we observe that especially *no-size* samples a low number of difficult objects. This might be an indication that it indeed mainly samples large objects, as large objects are typically not difficult.

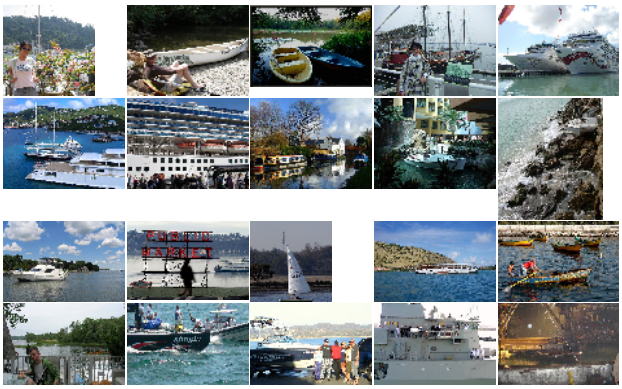
5.3.2 Case Studies: Selected Images

In this section we consider two case studies of which we qualitatively interpret the images selected. In the first case study we look at the first four active learning iterations of the *boat* category in the two-class setup. We consider the samples selected by pre-NMS averaging, which performs particularly well in this setup, HDBSCAN, which performs particularly bad and random sampling, which serves as a baseline. The results can be found in Figure 5.10a, Figure 5.10b and Figure 5.10c for the randomly selected samples, the samples selected by HDBSCAN and the samples selected by pre-NMS averaging. Moreover, in Figure 5.11 we show the corresponding difficult versus non-difficult image distributions as annotated in the Pascal VOC dataset [17].

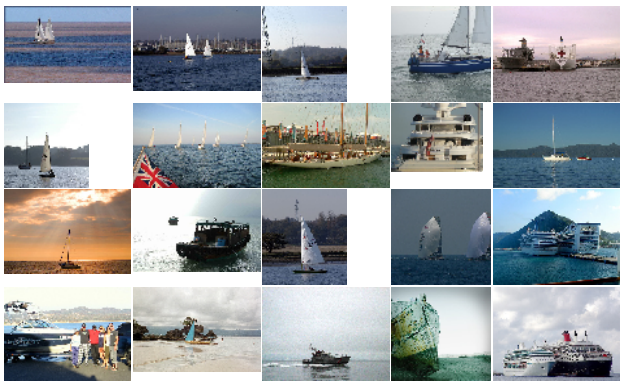
(a) Samples selected using random sampling in the first 4 iterations of the two-class experiments with image budget. Each row corresponds to an active learning iteration.



(b) Samples selected using HDBSCAN in the first 4 iterations of the two-class experiments with image budget. Each row corresponds to an active learning iteration.



(c) Samples selected using pre-NMS averaging in the first 4 iterations of the two-class experiments with image budget. Each row corresponds to an active learning iteration.



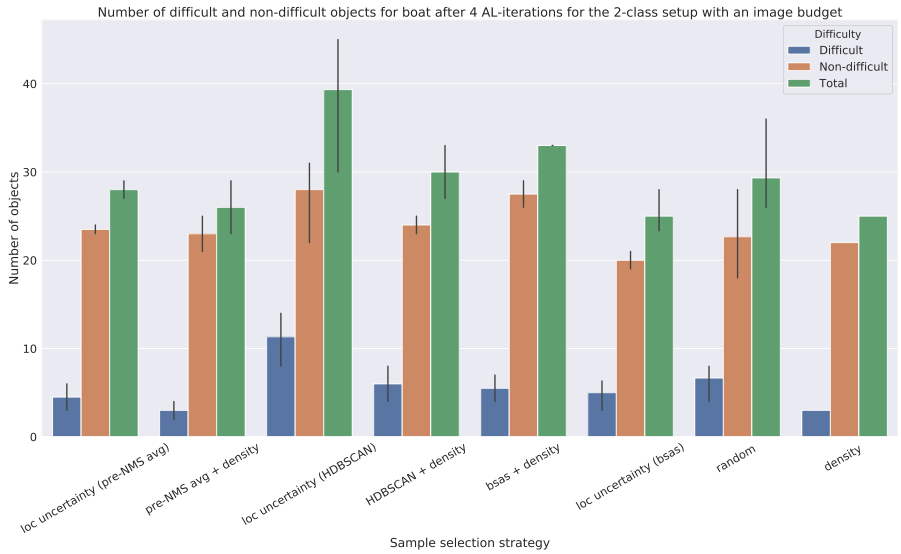


Figure 5.11: The distribution of difficult versus non-difficult sampled objects of the boat object category per sample selection strategy in the two-class setup with an image budget after all 4 active learning iterations. The objects from the seed-set are not included.

In the images selected with random sample selection a variety of boats is present. They are in general more isolated than the ones sampled using HDBSCAN. Moreover, there is less clutter from other objects in the pictures compared to the ones selected with HDBSCAN. In the samples selected using the HDBSCAN merging strategy, again a large variety of boats are present. However, often only a part of the boat is visible. In these cases it is harder to recognize it as a boat. For example, the second image on the second row displays a large passenger ship with small windows. The images are also cluttered, displaying many other objects in the pictures. On some of these images, such as the fourth image in the second row of Figure 5.10b, it is even hard for a human to directly observe the boat. Finally, in the samples selected by pre-NMS averaging we see that, overall, most boats are clearly boats. In most pictures there are no other objects and the boat is photographed as a whole. Therefore it might be easier to learn distinctive features of a boat from these images. Also, many boats are depicted from the side and have a sail. Many of the images seem typical depictions of boats. There are a few harder examples though, especially in the bottom row of Figure 5.11, the first, fourth and fifth image. This is also the iteration that resulted in a strong improvement compared to the other sample selection strategies Figure 5.3. In Figure 5.11, we observe that, according to PASCAL VOC, HDBSCAN indeed samples the hardest boat-objects, followed by random and then pre-NMS averaging.

In the second case we look at 25 images selected in the fourth iteration of the ablation study. We consider the ablation study for an qualitative analysis to gain more insights into the various components of our proposed localization uncertainty metric. We consider the fourth iteration as then we observe some notable divergences in performance for some of the sample selection metrics. Specifically, *no-size* starts to perform well on *potted plant* and *no-foreground-prob* starts to perform better on *boat*.

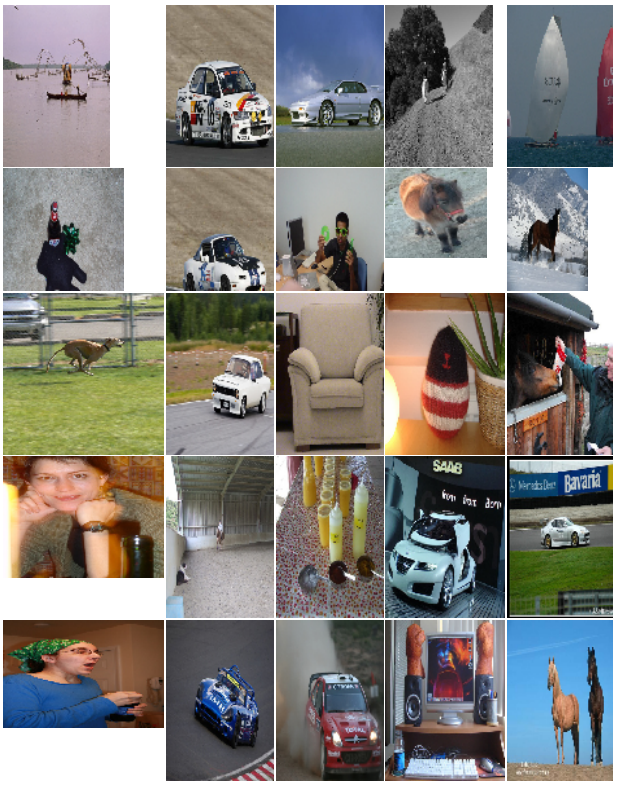


Figure 5.12: The samples selected in the fourth active learning iteration of the multi-class setup with localization uncertainty without the scaling by the predicted foreground probabilities. For these images we decided to resize the images because otherwise the bottle on the images in the first column at the bottom two rows was not clearly depicted.

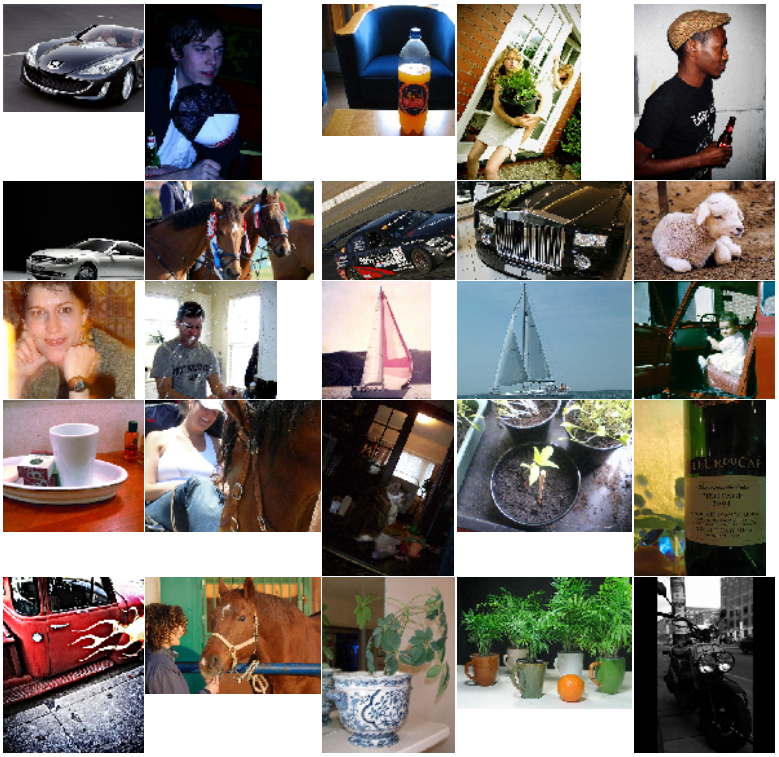


Figure 5.13: The samples selected in the fourth active learning iteration of the multi-class setup with localization uncertainty without the division by the size of the probabilistic object detections.

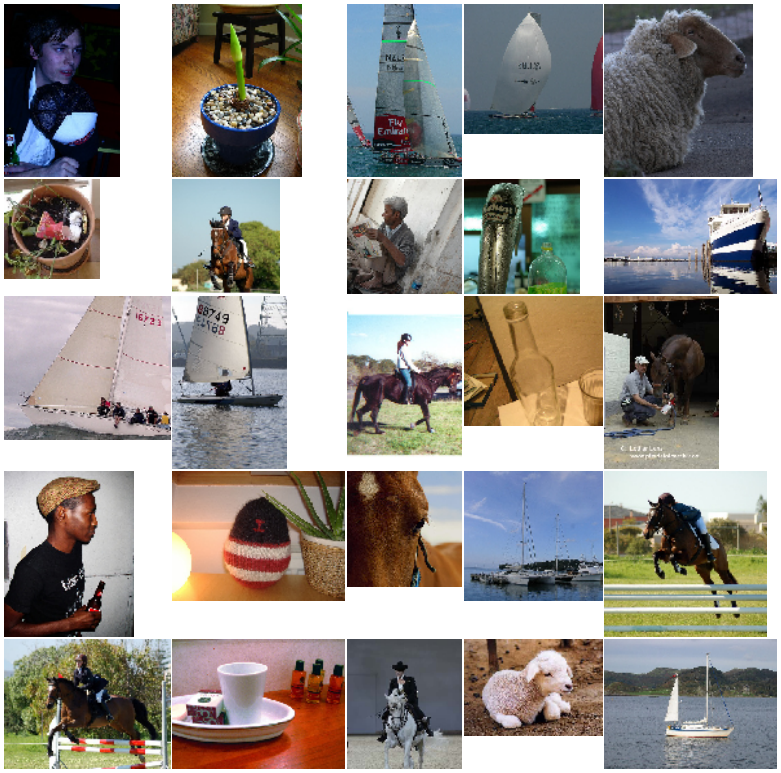


Figure 5.14: The samples selected in the fourth active learning iteration of the multi-class setup with our localization uncertainty metric.

In Figure 5.12 samples selected *no-foreground-prob*, are depicted. We do not observe any noteworthy images or patterns. In Figure 5.13 the samples selected by *no-size* are depicted. As we assumed, almost all objects are large objects compared to when we scale by the object size in *no-foreground-prob* and *loc uncertainty*. We have a qualitative explanation for the relative performance gain of *no-foreground-prob* on *boat*. The one image containing boats is probably not the reason that *no-foreground-prob* starts to perform relatively better. Instead, *no-size*, the sampling strategy that was closest in terms of performance, sampled a sailing yacht that has a pinkish glow. As typically boat images are blueish or greyish, this might have hurt performance. Regarding the fact that *no-size* starts to perform better on *potted plant*, we conjecture this is because it samples relatively many of them.

Chapter 6

Conclusion

In this thesis we proposed a new sample selection metric that incorporates localization uncertainty obtained through sampling-based probabilistic bounding boxes $\mathcal{O}$. Specifically, for each $\mathcal{O}$ we sum the traces of the covariance matrices of the two multivariate Gaussian distributions that define the probabilistic bounding boxes to obtain a localization uncertainty. In our localization uncertainty metric we divide this sum by the size of the predicted bounding box and scaled them by the predicted probability that it is a foreground object. Using this metric, we showed promising results on a subset of six carefully selected classes of the Pascal VOC 2007 dataset in both a multi-class as well as a two-class setup. Our metric outperformed a range of other sample selection strategies, including density-based sampling, classification uncertainty-based sampling and sampling at random. We showed that the most beneficial way of obtaining these probabilistic bounding boxes is with pre-NMS averaging. Interestingly, the performance of our localization uncertainty metric with the other two merging strategies (HDBSCAN and BSAS) was substantially worse. The way in which the localization uncertainty is obtained is thus very important. In an ablation study we showed that when not scaling by either the foreground probability or size of the object, the localization uncertainty alone already provides valuable information regarding the informativeness of objects. Without the scaling parameters it outperformed the other sampling strategies as well, however, in most cases our localization uncertainty metric including both scaling variables performed best out of all sampling strategies.

We also combined our localization uncertainty metric with a density sampling metric. This shows us that, for the HDBSCAN and BSAS merging strategies, our localization uncertainty metric appears to sample from low-density regions of the image space. Pre-NMS averaging was the only merging strategy that did not benefit from adding density sampling and therefore probably does not suffer from sampling from low-density regions of the image space.

Furthermore, we combined our localization uncertainty metric with a classification uncertainty metric, because object detection consists of both localizing as well as classifying what is on an image. We observed that combining the two is typically worse than using either of the two. Therefore, we conclude that our localization and classification uncertainty metrics were not complementary to each other.

Finally, we show in a qualitative study that even though we refer to our metric as a localization uncertainty metric, it does not necessarily select the most difficult objects. This is an important insight, as it suggests that it is not necessarily good to sample objects that are difficult, but that the localization *'information'* that can be obtained trough probabilistic bounding boxes, is valuable for active learning nonetheless.

Chapter 7

Discussion and Future work

In this work we show promising results for our newly proposed localization uncertainty metric with one of the merging strategies. Note however, that the other two methods, HDBSCAN and BSAS merging, are more susceptible to hyperparameter tuning and we have not done a hyperparameter search. Their performance might have been better with other hyperparameters. For example, in early active learning iterations, the detectors' performance is not very good yet. For BSAS merging it might be beneficial to set a low IoU threshold in early rounds and gradually increase this. Nevertheless, in active learning, there is not a lot of data to perform hyperparameterization on, making pre-NMS averaging an even more suitable merging method compared to the other two. As pre-NMS averaging is the best merging strategy, other forms of NMS such as soft-NMS [10] might be considered in future research.

A major drawback of our method is the reliance on sampling-based object detectors. The computational cost could therefore outweigh the savings that are obtained by selecting more informative samples. Furthermore, in case one does not have a distributed computing setup, the time between active learning iterations also increases. This makes the method less interactive and could reduce its practicality. Further research might look into using different, cheaper backbones for the neural network during sample selection, as 80% [46] of the time spent in the forward pass of the SSD consists of the VGG-16 backbone.

Though we use an image-level sample definition, further research can also look into using the proposed localization uncertainty with an object-level definition. Another promising avenue of research would be to use other ways of obtaining probabilistic object detection, in conjunction with our localization uncertainty metric. As the field of probabilistic object detection is relatively new, we expect that with development of future methods, the benefits of the proposed localization uncertainty metric is likely to increase.

Since in our multi-class experiments, the density metric mainly sampled from a few object categories, further research might look into density metrics that are better suited for multi-class sample selection. Inspiration for this can be found in the work by Rudinac, Martha and Hanjalic [63], in which they experiment with various ways of obtaining representativeness scores using the predicted classes on each frame for text-based video retrieval.

As we only show results on a subset of a single dataset, further research must be done to research the generalizability of our method. Furthermore, we used a fairly small part of the total unlabeled set in our experiments, only mimicking the early part of the active learning process. It might be that our method selects samples that are highly informative in early rounds, but not anymore in later rounds when the detectors performance is already fairly good.

Finally, in this research we let each image in the unlabeled data pool have at least one relevant object. Future research must be conducted for settings where this is not the case. We expect that with a sufficient seed set,

the uncertainty metrics, both for localization and classification uncertainty, might still perform relatively well out of the box due to the weighting of each observation by the probability of it being a foreground object.

Bibliography

[1] Pascal voc 2007 object detection page. http://host.robots.ox.ac.uk/pascal/VOC/voc2007/htmldoc/voc.html#SECTION00050000000000000000, 2007. Accessed: 2019-10-11.

[2] Anderson, C. H., Van Essen, D. C., and Olshausen, B. A. Directed visual attention and the dynamic control of information flow. In *Neurobiology of attention*. Elsevier, 2005, pp. 11–17.

[3] Arganda-Carreras, I., Turaga, S. C., Berger, D. R., Cireşan, D., Giusti, A., Gambardella, L. M., Schmidhuber, J., Laptev, D., Dwivedi, S., Buhmann, J. M., et al. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in neuroanatomy 9* (2015), 142.

[4] Atlas, L. E., Cohn, D. A., and Ladner, R. E. Training connectionist networks with queries and selective sampling. In *Advances in neural information processing systems* (1990), pp. 566–573.

[5] Babenko, A., and Lempitsky, V. Aggregating local deep features for image retrieval. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1269–1277.

[6] Babenko, A., Slesarev, A., Chigorin, A., and Lempitsky, V. Neural codes for image retrieval. In *European conference on computer vision* (2014), Springer, pp. 584–599.

[7] Bappy, J. H., Paul, S., Tuncel, E., and Roy-Chowdhury, A. K. The impact of typicality for informative representative selection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 5878–5887.

[8] Beluch, W. H., Genewein, T., Nürnberger, A., and Köhler, J. M. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 9368–9377.

[9] Bo, L., and Sminchisescu, C. Efficient match kernel between sets of features for visual recognition. In *Advances in neural information processing systems* (2009), pp. 135–143.

[10] Bodla, N., Singh, B., Chellappa, R., and Davis, L. S. Soft-nms–improving object detection with one line of code. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 5561–5569.

[11] Bolles, R. C., and Fischler, M. A. A ransac-based approach to model fitting and its application to finding cylinders in range data. In *IJCAI* (1981), vol. 1981, pp. 637–643.

[12] Campello, R. J., Moulavi, D., and Sander, J. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining* (2013), Springer, pp. 160–172.

[13] Campello, R. J., Moulavi, D., Zimek, A., and Sander, J. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD) 10*, 1 (2015), 5.

[14] Cohn, D., Atlas, L., and Ladner, R. Improving generalization with active learning. *Machine learning 15*, 2 (1994), 201–221.

[15] Cohn, D. A., Ghahramani, Z., and Jordan, M. I. Active learning with statistical models. *Journal of artificial intelligence research 4* (1996), 129–145.

[16] Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D. Scalable object detection using deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), pp. 2147–2154.

[17] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *International journal of computer vision 88*, 2 (2010), 303–338.

[18] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[19] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence 32*, 9 (2009), 1627–1645.

[20] Gal, Y. Uncertainty in deep learning. *University of Cambridge* (2016).

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

[22] Gal, Y., Islam, R., and Ghahramani, Z. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), JMLR. org, pp. 1183–1192.

[23] Girshick, R. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1440–1448.

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

[25] Glorot, X., and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010), pp. 249–256.

[26] Gong, Y., Wang, L., Guo, R., and Lazebnik, S. Multi-scale orderless pooling of deep convolutional activation features. In *European conference on computer vision* (2014), Springer, pp. 392–407.

[27] Hall, D., Dayoub, F., Skinner, J., Corke, P., Carneiro, G., and Sünderhauf, N. Probability-based detection quality (pdq): A probabilistic approach to detection evaluation. *arXiv preprint arXiv:1811.10800* (2018).

[28] He, K., Zhang, X., Ren, S., and Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence 37*, 9 (2015), 1904–1916.

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

[30] He, Y., Li, B., and Zhao, Y. New default box strategy of ssd for small target detection. In *International Conference on Neural Information Processing* (2018), Springer, pp. 416–425.

[31] He, Y., Zhu, C., Wang, J., Savvides, M., and Zhang, X. Bounding box regression with uncertainty for accurate object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 2888–2897.

[32] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012).

[33] Holschneider, M., Kronland-Martinet, R., Morlet, J., and Tchamitchian, P. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*. Springer, 1990, pp. 286–297.

[34] Hosang, J., Benenson, R., Dollár, P., and Schiele, B. What makes for effective detection proposals? *IEEE transactions on pattern analysis and machine intelligence 38*, 4 (2016), 814–830.

[35] Howard, A. G. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402* (2013).

[36] Huang, T. S., Dagli, C. K., Rajaram, S., Chang, E. Y., Mandel, M. I., Poliner, G. E., and Ellis, D. P. Active learning for interactive multimedia retrieval. *Proceedings of the IEEE 96*, 4 (2008), 648–667.

[37] Huijser, M., and van Gemert, J. C. Active decision boundary annotation with deep generative models. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 5286–5295.

[38] Jégou, H., and Zisserman, A. Triangulation embedding and democratic aggregation for image search. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), pp. 3310–3317.

[39] Kao, C.-C., Lee, T.-Y., Sen, P., and Liu, M.-Y. Localization-aware active learning for object detection. In *Asian Conference on Computer Vision* (2018), Springer, pp. 506–522.

[40] Kornblith, S., Shlens, J., and Le, Q. V. Do better imagenet models transfer better? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 2661–2671.

[41] Kuhn, H. W. The hungarian method for the assignment problem. *Naval research logistics quarterly 2*, 1-2 (1955), 83–97.

[42] Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems* (2017), pp. 6402–6413.

[43] Li, X., and Guo, Y. Adaptive active learning for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 859–866.

[44] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence* (2018).

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

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

[47] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision 60*, 2 (2004), 91–110.

[48] Marr, D. Vision: A computational investigation into the human representation and processing of visual information.

[49] McInnes, L., Healy, J., and Astels, S. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software 2*, 11 (mar 2017).

[50] Miller, D., Dayoub, F., Milford, M., and Sünderhauf, N. Evaluating merging strategies for sampling-based uncertainty techniques in object detection. *arXiv preprint arXiv:1809.06006* (2018).

[51] Miller, D., Nicholson, L., Dayoub, F., and Sünderhauf, N. Dropout sampling for robust object detection in open-set conditions. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), IEEE, pp. 1–7.

[52] Miller, D., Sünderhauf, N., Zhang, H., Hall, D., and Dayoub, F. Benchmarking sampling-based probabilistic object detectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2019), pp. 42–45.

[53] Nguyen, H. T., and Smeulders, A. Active learning using pre-clustering. In *Proceedings of the twenty-first international conference on Machine learning* (2004), ACM, p. 79.

[54] Oh, S., Hoogs, A., Perera, A., Cuntoor, N., Chen, C.-C., Lee, J. T., Mukherjee, S., Aggarwal, J., Lee, H., Davis, L., et al. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR 2011* (2011), IEEE, pp. 3153–3160.

[55] Oksuz, K., Can Cam, B., Akbas, E., and Kalkan, S. Localization recall precision (lrp): A new performance metric for object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 504–519.

[56] Osband, I. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *NIPS Workshop on Bayesian Deep Learning* (2016).

[57] Perronnin, F., Liu, Y., Sánchez, J., and Poirier, H. Large-scale image retrieval with compressed fisher vectors. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2010), IEEE, pp. 3384–3391.

[58] Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. Lost in quantization: Improving particular object retrieval in large scale image databases. In *IEEE Conference on Computer Vision and Pattern Recognition* (2008).

[59] Radenović, F., Iscen, A., Tolias, G., Avrithis, Y., and Chum, O. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 5706–5715.

[60] Radenović, F., Tolias, G., and Chum, O. Cnn image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In *European conference on computer vision* (2016), Springer, pp. 3–20.

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

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

[63] Rudinac, S., Larson, M., and Hanjalic, A. Leveraging visual concepts and query performance prediction for semantic-theme-based video retrieval. *International Journal of Multimedia Information Retrieval 1*, 4 (Dec 2012), 263–280.

[64] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision 115*, 3 (2015), 211–252.

[65] Russakovsky, O., Li, L.-J., and Fei-Fei, L. Best of both worlds: human-machine collaboration for object annotation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 2121–2131.

[66] Seddati, O., Dupont, S., Mahmoudi, S., and Parian, M. Towards good practices for image retrieval based on cnn features. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 1246–1255.

[67] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229* (2013).

[68] Settles, B. Active learning literature survey. 2010. *Computer Sciences Technical Report 1648* (2014).

[69] Settles, B., and Craven, M. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing* (2008), Association for Computational Linguistics, pp. 1070–1079.

[70] Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (2014), pp. 806–813.

[71] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[72] Sirinukunwattana, K., Pluim, J. P., Chen, H., Qi, X., Heng, P.-A., Guo, Y. B., Wang, L. Y., Matuszewski, B. J., Bruni, E., Sanchez, U., et al. Gland segmentation in colon histology images: The glas challenge contest. *Medical image analysis 35* (2017), 489–502.

[73] Sivaraman, S., and Trivedi, M. M. Active learning for on-road vehicle detection: A comparative study. *Machine vision and applications 25*, 3 (2014), 599–611.

[74] Skinner, J., Hall, D., Zhang, H., Dayoub, F., and Sünderhauf, N. The probabilistic object detection challenge. *arXiv preprint arXiv:1903.07840* (2019).

[75] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research 15*, 1 (2014), 1929–1958.

[76] Szeliski, R. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[77] Torralba, A., Efros, A. A., et al. Unbiased look at dataset bias. In *CVPR* (2011), vol. 1, Citeseer, p. 7.

[78] Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. Selective search for object recognition. *International journal of computer vision 104*, 2 (2013), 154–171.

[79] Vijayanarasimhan, S., and Grauman, K. Large-scale live active learning: Training object detectors with crawled data and crowds. *International Journal of Computer Vision 108*, 1-2 (2014), 97–114.

[80] Wang, K., Lin, L., Yan, X., Chen, Z., Zhang, D., and Zhang, L. Cost-effective object detection: Active sample mining with switchable selection criteria. *IEEE Transactions on Neural Networks and Learning Systems*, 99 (2018), 1–17.

[81] Wang, K., Yan, X., Zhang, D., Zhang, L., and Lin, L. Towards human-machine cooperation: Self-supervised sample mining for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 1605–1613.

[82] Wang, K., Zhang, D., Li, Y., Zhang, R., and Lin, L. Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology 27*, 12 (2017), 2591–2600.

[83] Wang, M., and Hua, X.-S. Active learning in multimedia annotation and retrieval: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST) 2*, 2 (2011), 10.

[84] Yang, L., Zhang, Y., Chen, J., Zhang, S., and Chen, D. Z. Suggestive annotation: A deep active learning framework for biomedical image segmentation. In *International conference on medical image computing and computer-assisted intervention* (2017), Springer, pp. 399–407.

[85] Yang, Y., Ma, Z., Nie, F., Chang, X., and Hauptmann, A. G. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision 113*, 2 (2015), 113–127.

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

[87] Zahlka, J., Rudinac, S., Jnsson, B. ., Koelma, D. C., and Worring, M. Blackthorn: Large-scale interactive multimodal learning. *IEEE Transactions on Multimedia 20*, 3 (March 2018), 687–698.

[88] Zhao, Z.-Q., Zheng, P., Xu, S.-t., and Wu, X. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems* (2019).

Appendix

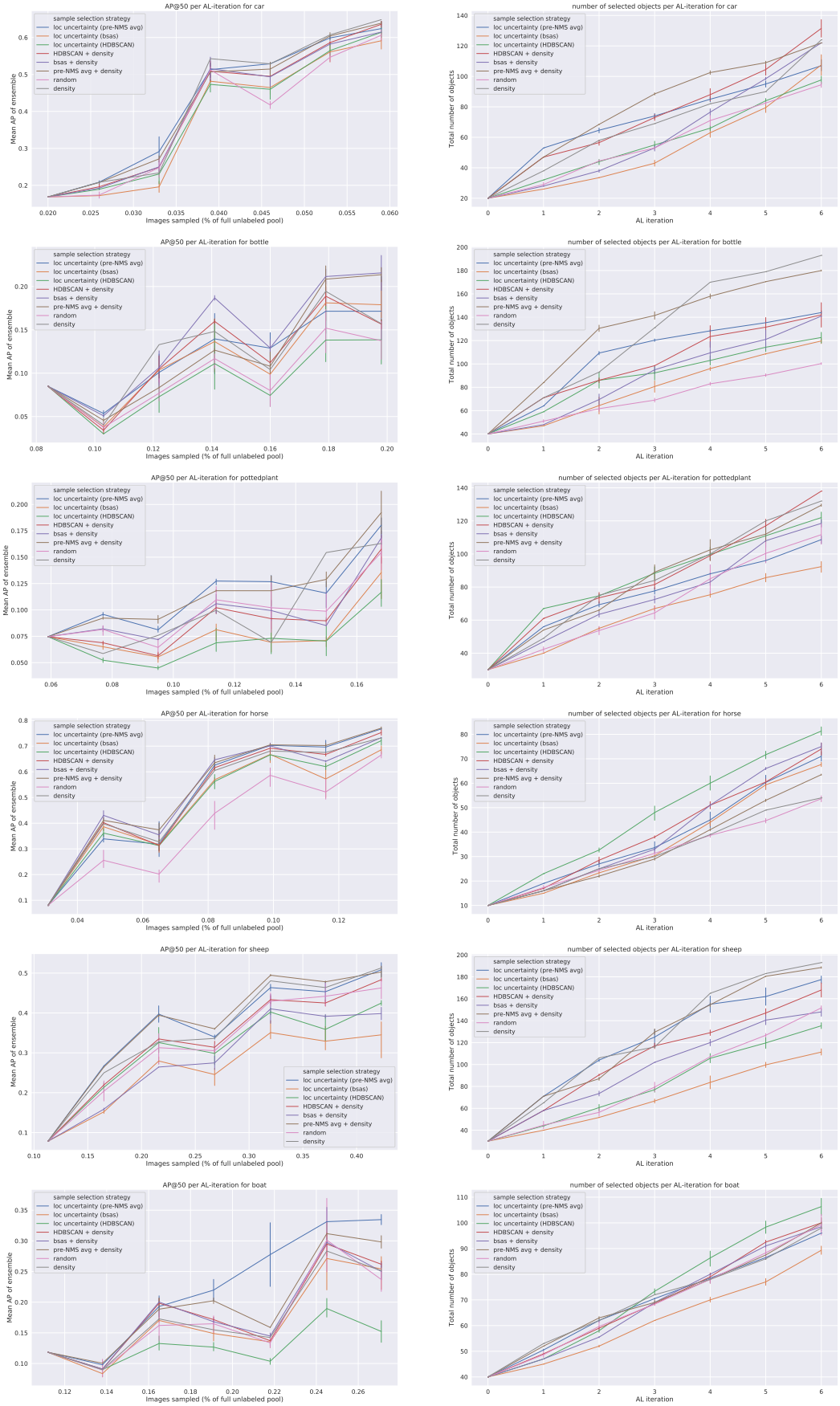


Figure 7.1: All results of the image level of the two-class setup. The left column is AP averaged over the 6 SSDs and the right column shows the number of images selected for each iteration. Each row shows the results of a different object category. The plots are interpolated between each active learning iteration and the horizontal bars are confidence intervals on the performance and number of selected images per active learning iteration for the 3 runs per stochastic sample selection strategy. Each round 5 images are sampled. Note that the axes do not start at the origin.

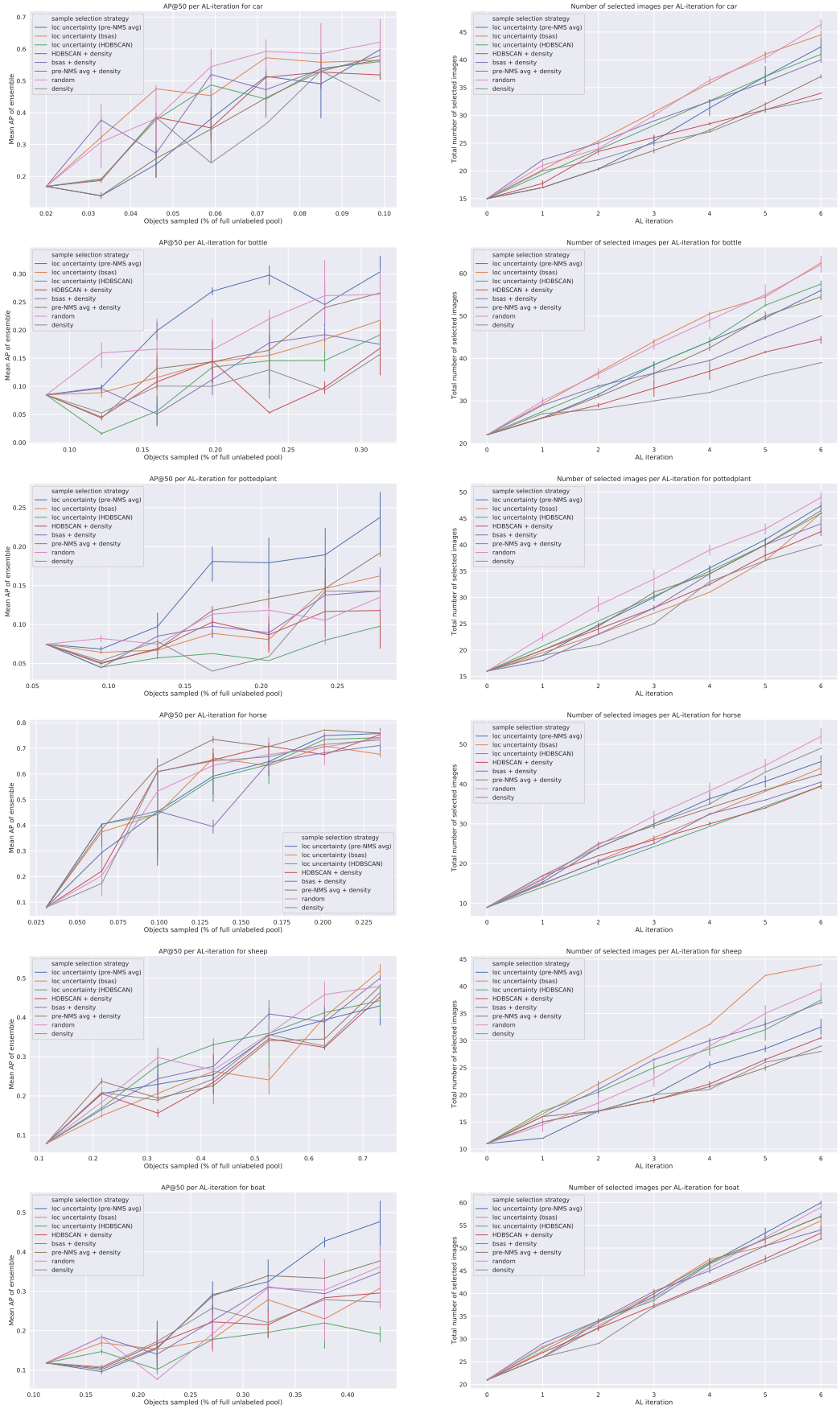


Figure 7.2: All results of the object level of the two-class setup. The left column is AP averaged over the 6 SSDs and the right column shows the number of objects selected for each iteration. Each row shows the results of a different object category. The plots are interpolated between each active learning iteration and the horizontal bars are confidence intervals on the performance and number of selected images per active learning iteration for the 3 runs per stochastic sample selection strategy. Each round 10 objects are sampled. Note that the axes do not start at the origin.

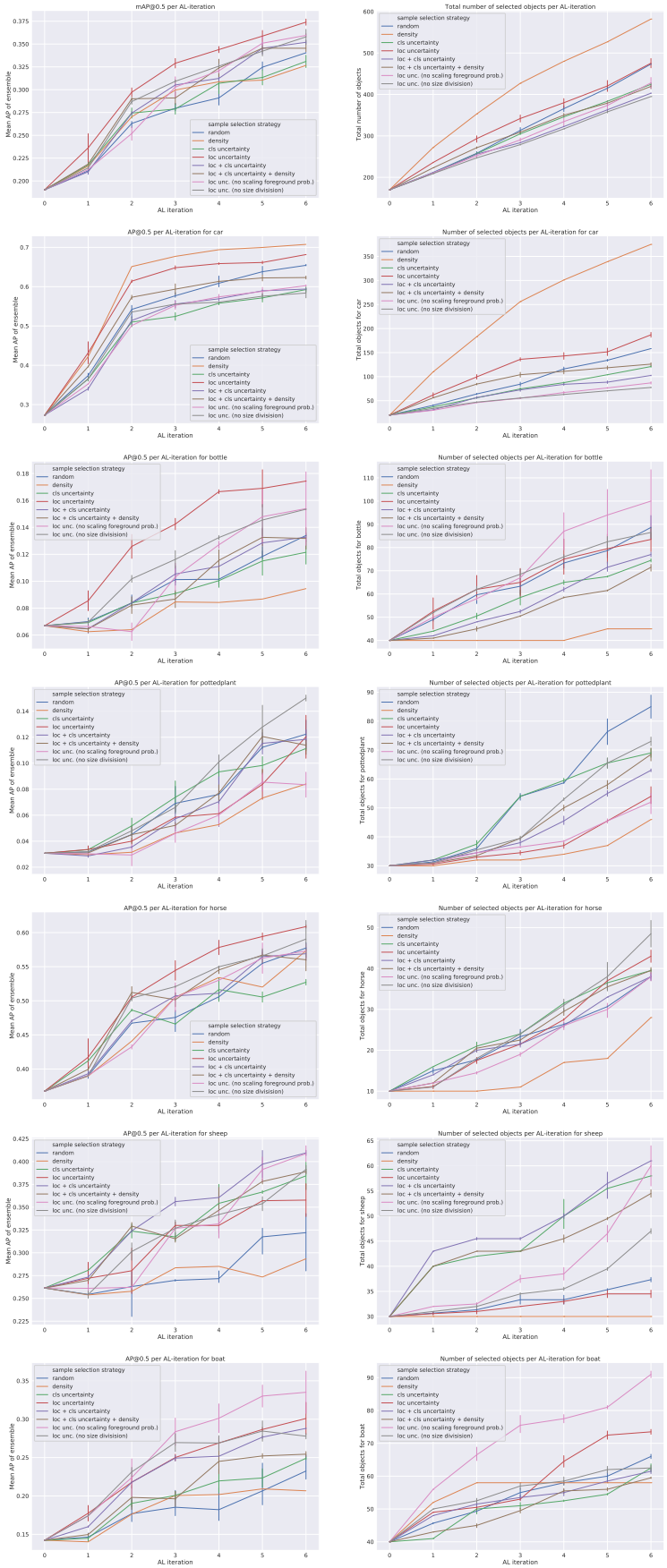


Figure 7.3: All results of the multi-class setup, except for the performance when trained on the full-dataset. The left column is the AP averaged over the 6 SSDs in the ensemble with the confidence bounds from the two runs per method. In the right column we show the number of objects that are sampled from the labeled pool for each class for each sample selection method. Each round 25 images are selected. Note that the axes do not start at the origin.

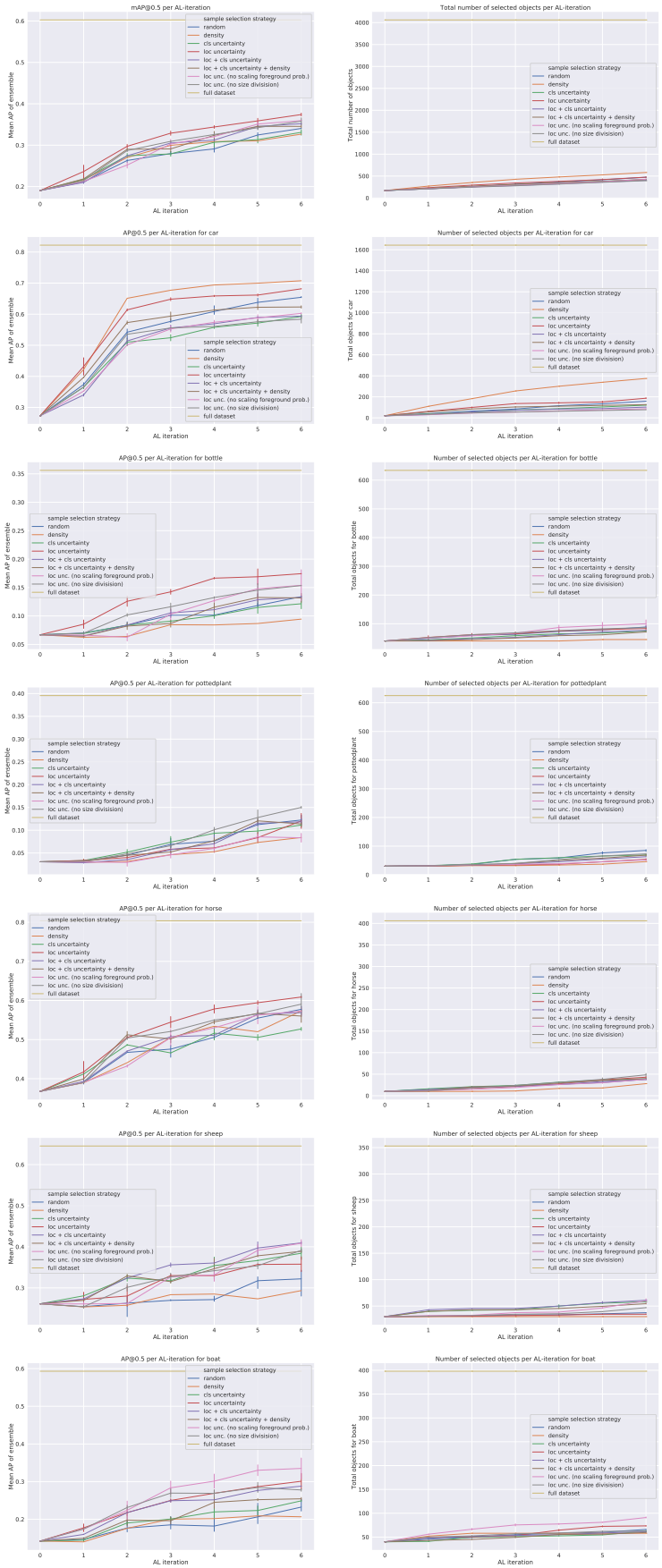


Figure 7.4: All results of the multi-class setup, including performance when trained on the full-dataset. The left column is the AP averaged over the 6 SSDs in the ensemble with the confidence bounds from the two runs per method. In the right column we show the number of objects that are sampled from the labeled pool for each class for each sample selection method. Each round 25 images are selected. Note that the axes do not start at the origin.

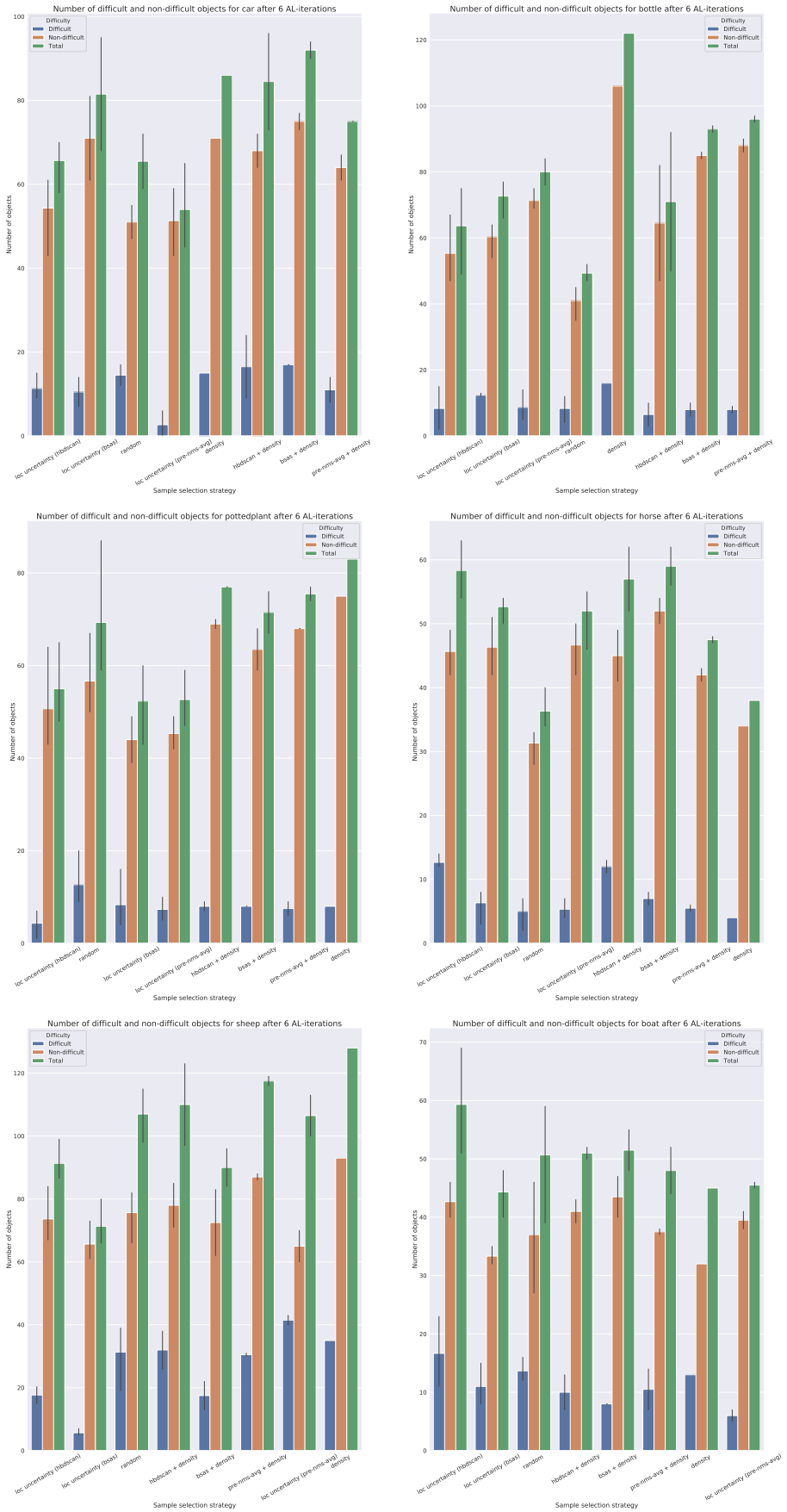


Figure 7.5: The distributions of difficult versus non-difficult samples for all classes and all sample selection strategies in the multi-class setup after 6 active learning iterations. The seed set is not included.

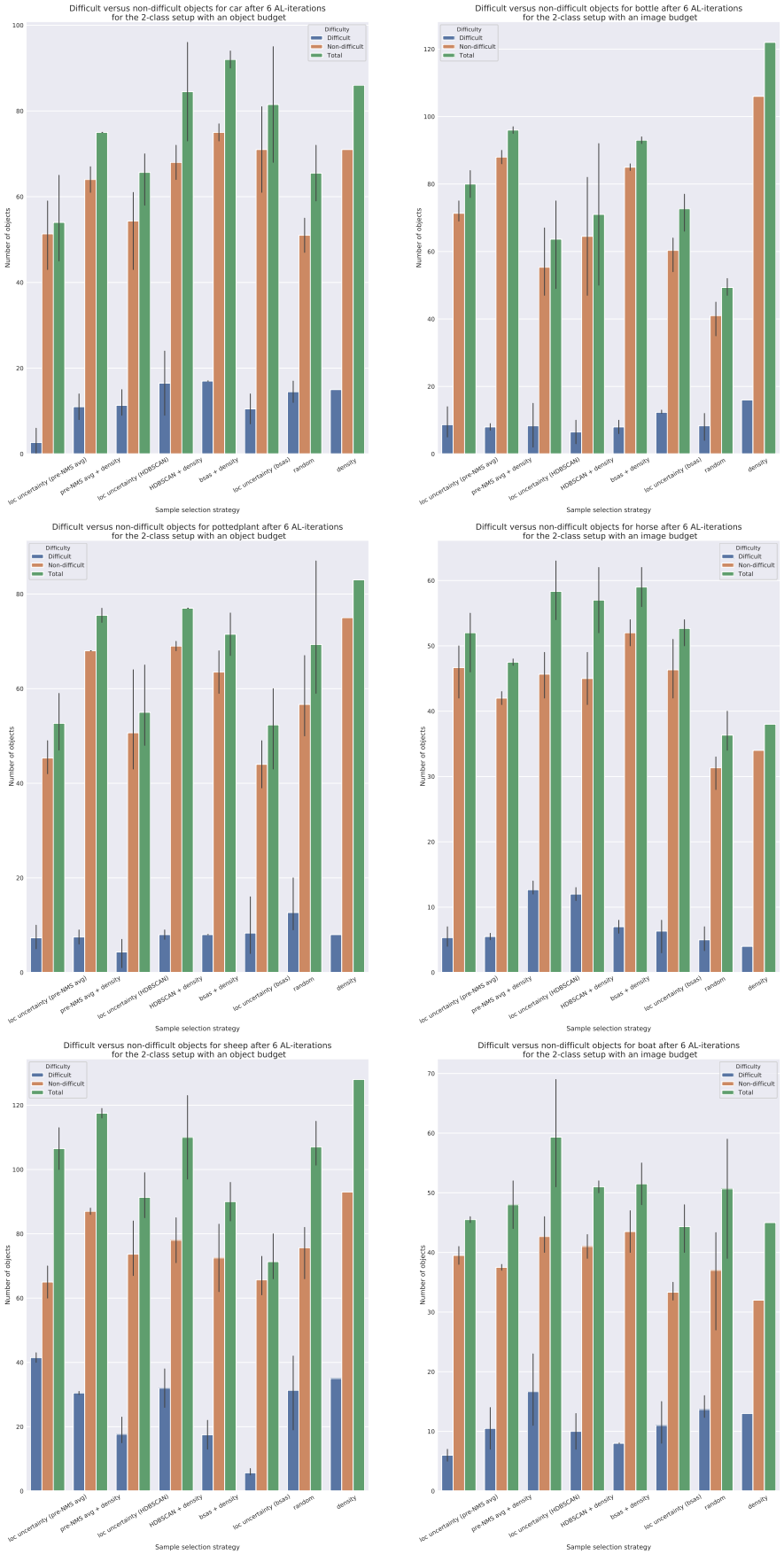


Figure 7.6: The distributions of difficult versus non-difficult samples for all classes and all sample selection strategies in the two-class setup with image budget definition after 6 active learning iterations. The seed set is not included.

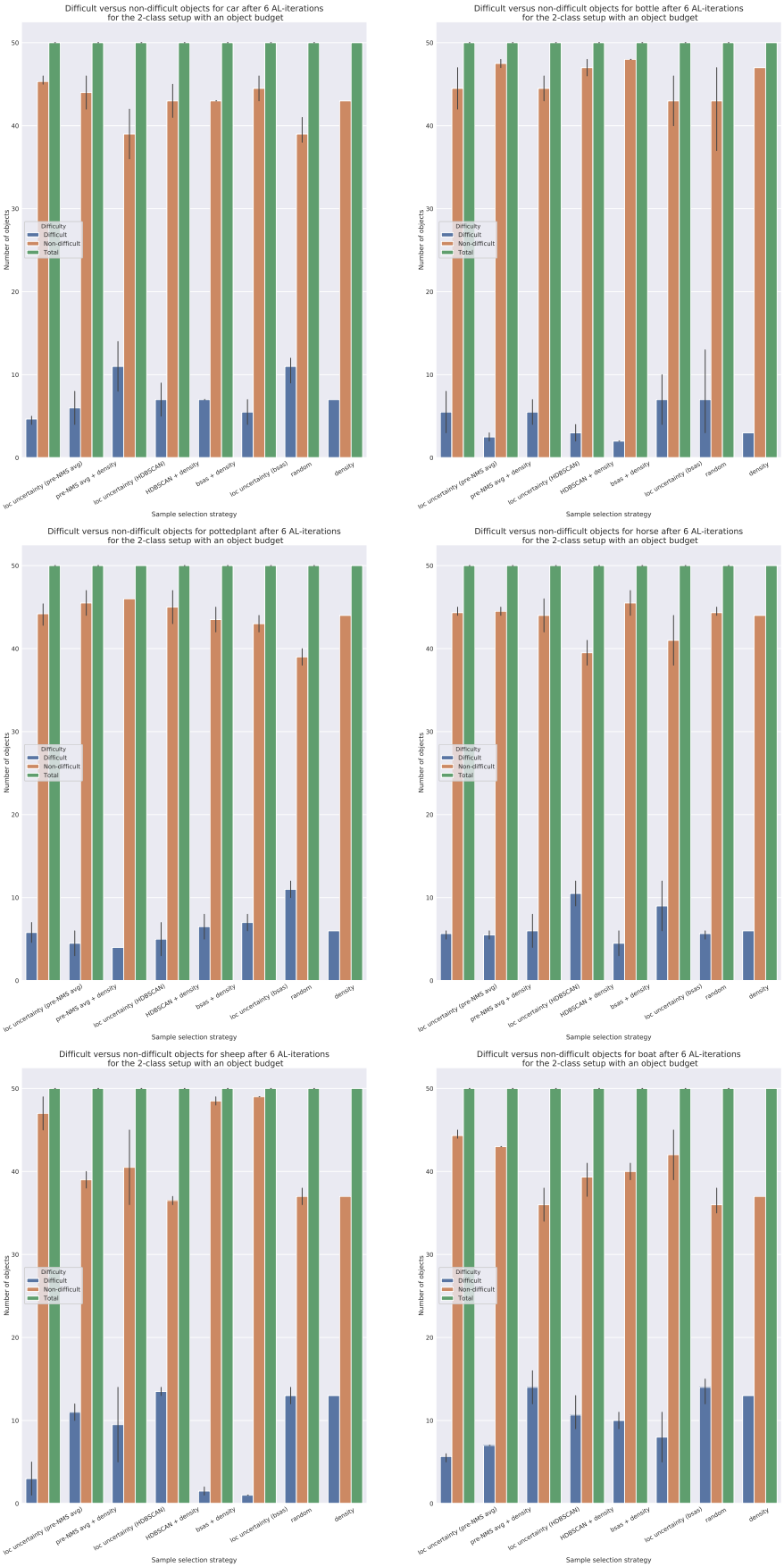


Figure 7.7: The distributions of difficult versus non-difficult samples for all classes and all sample selection strategies in the two-class setup with object budget definition after 6 active learning iterations. The seed set is not included.