CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Importance Sampling in Deep Learning Object Detection

An empirical investigation into accelerating deep learning object detection training by exploiting informative data points

Master's thesis in Computer science and engineering

Alexander Huang, Johannes Kvernes

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

Importance Sampling in Deep Learning Object Detection

An empirical investigation into accelerating deep learning object detection training by exploiting informative data points

Alexander Huang & Johannes Kvernes

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

Importance Sampling in Deep Learning Object Detection
An empirical investigation into accelerating deep learning object detection training by exploiting informative data points
Alexander Huang, Johannes Kvernes

Acedmic supervisor: Yinan Yu, Department of Computer Science and Engineering
Industrial supervisors: Olle Månsson, Zenseact. Willem Verbeke, Zenseact
Examiner: Krasimir Angelov, Department of Computer Science and Engineering

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2023

Importance Sampling in Deep Learning Object Detection
An empirical investigation into accelerating deep learning object detection training by exploiting informative data points
Alexander Huang
Johannes Kvernes
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In recent years, the field of deep learning object detection has witnessed a notable surge in progress, largely fueled by the growth of available data. While the ever growing amount of data has enabled complex object detection models to improve generalization performance and to facilitate training phases that are less prone to over-fitting, the time to reach desired performance with this remarkable amount of data has been reported to be an emerging problem in practice. This is certainly the case when each sample consist of high resolution image data that could burden the capacity of loading data.

This thesis explores the possibilities of leveraging importance sampling as a means to accelerate gradient-based training of an object detection model. This avenue of research aims at biasing the sampling of training examples during training, in the hope of exploiting informative samples and reduce the amount of computation on uninformative, noisy and out-of-distribution samples. While previous art shows compelling evidence of importance sampling in deep learning, it is consistently reported in a context of less complex tasks. In this work, we propose techniques that can be applied to single-stage anchor-free object detectors, in order to adopt importance sampling to accelerate training. Our methods do not require modifications to the objective function, and allow for a biased sampling procedure that remains consistent across runs and samples.

Our results suggest that an uncertainty-based heuristic outperforms loss-based heuristics, and that object detection training can be subject to a remarkable speed-up in terms of reaching the baseline's performance in fewer iterations, where the baseline samples the training examples uniformly without replacement. Furthermore, the empiric observations reported in this work also indicate that an increased final generalization performance can be achieved given an equal amount of training time when compared to the baseline.

Keywords: Importance sampling, deep learning, object detection, accelerate training, sampling heuristic, gradient descent, loss, uncertainty, threshold-closeness.

Acknowledgements

We would like to show our greatest appreciation to Zenseact for the opportunity to work with an interesting topic and a fantastic team, and for lending us the equipment and computational resources that have made this project possible.

Additionally, we would like to thank our industrial supervisors, Willem Verbeke and Olle Månsson, whose guidance and mentorship have been instrumental in shaping this project. They have, with great expertise and sheer commitment to this thesis, played a crucial role in resolving complex problems and equipped us with knowledge that has not only enriched the project, but also ourselves as future practitioners.

Lastly, we would like to extend our gratitude to our academic supervisor, Yinan Yu, who has been active throughout the project, making the academic formalities a smooth process as well as providing us with valuable insights and lead-way into the cutting-edge research through relevant literature.

Alexander Huang, Gothenburg, 2023-06-13

Johannes Kvernes, Gothenburg, 2023-06-13

Contents

List of Acronyms

AD Autonomous Driving.

ADAS Advanced Driver Assistance Systems.

AP Average Precision.

CNN Convolutional Neural Network.

CPU Central Processing Unit.

DL Deep Learning.

DLA Deep Layer Aggregation.

FCN Fully Convolutional Network.

FCOS Fully Convolutional One-Stage Object Detector.

FN False Negative.

FP False Positive.

FPN Feature Pyramid Network.

IoU Intersection over Union.

IS Importance Sampling.

LIS Loss Importance Sampler.

LIS-IR Loss Importance Sampler with Ignored Regression.

LIS-RB Loss Importance Sampler with Rejection-based sampling.

LIS-RHO Loss Importance Sampler with Reducible Hold Out framework.

mAP Mean Average Precision.

ML Machine Learning.

MLP Multilayer Perceptron.

NMS Non-Maximum Suppression.

OD Object Detection.

RHO Reducible Hold-out.

RoI Region of interest.

SSD Single-Stage Detector.

TN True Negative.

TP True Positive.

UIS Uncertainty Importance Sampler.

UIS-IR Uncertainty Importance Sampler with Ignored Regression.

UIS-RB Uncertainty Importance Sampler with Rejection-based sampling.

ZOD Zenseact Open Dataset.

1

Introduction

The remarkable increase in available data has enabled rapid development of deep neural networks. Various highly complex deep neural network models have been shown to outperform other methods in a plethora of tasks in fields such as computer vision and natural language processing. However, big datasets and complex models are generally mirrored by a substantial computational cost that entails a central challenge when training these types of machine learning models, highlighting the importance of improving the efficiency of gradient-based training algorithms.

One aspect of streamlining training is to supervise the model with samples from training data with a frequency that is proportional to their respective importance score. For context, it has rendered itself obvious to many practitioners that not all samples are of equal difficulty for an Machine Learning (ML) model to solve. During the training phase, deep learning models tend to converge rapidly in the early iterations, whereas in the intermediate and later stages, a vast majority of samples become significantly easier for the model to predict correctly, resulting in minuscule gradient updates. Compellingly, a portion of samples could possibly be sampled less frequently, while other more informative samples could be exploited without compromising the performance of the final model. This seems to have the potential to accelerating training. This phenomenon has been showcased in numerous dissertations related to data pruning and core-set selection [1], [2], example difficulty [3]–[5], curriculum learning [6], and various other paradigms including Importance Sampling (IS) [7]–[12].

This thesis explores the application of various IS techniques as a means to accelerate training of deep learning Object Detection (OD) models. By leveraging IS, training can be concentrated on samples that tend to contribute more to the process of finding a local minimum in the optimization problem at hand, supposedly leading to fewer training iterations and/or better generalization performance in comparison to the standard procedure of uniform sampling without replacement. As such, training times would be less susceptible to an ever-growing amount of data and could, in practice, free up computational resources for other problems or accelerate research and development since the time garnered by lower training times could be allocated to conducting more experiments in a less or equal amount of time.

1.1 Purpose

The purpose of this thesis is to empirically investigate, develop and propose a representative approach for applying importance sampling in the task of OD, an application of importance sampling that has, as opposed to image classification and to our best knowledge at the time of writing, yet to be studied thoroughly in the current art. In our work, we anticipate to showcase examples of how importance sampling can be leveraged to accelerate the training of an OD model in a computationally constrained environment for a real-time application such as Autonomous Driving (AD). The main research questions for this thesis are the following:

1. **How do different types of importance sampling heuristics perform in the task of deep learning object detection?**
2. **What are the effects of different sampling strategies when training a deep learning object detector, and how do they perform?**
3. **Can importance sampling be used to accelerate learning in low-density regions of a data distribution?**

1.2 Collaboration

This project was conducted at Zenseact, a technology company centered around developing AD and Advanced Driver Assistance Systems (ADAS) [13]. Zenseact thrives on recent advances in the field of machine learning and leverages R&D operations to provide state-of-the-art solutions for renowned vehicle manufacturers. Under those circumstances and from the outset, practitioners deemed it necessary to accelerate the training process of the in-house OD model, and as such, this project was conducted in close collaboration with our supervisors at the company. Before closing this paragraph, we want to highlight the rigorous work of which this thesis was extended from [14]. Similarly to us, Johansson and Lindberg [14] completed their project at Zenseact and investigated IS in the context of deep learning. The authors even touched on its application in training an object detector. Their study garnered a strong base of knowledge to proceed from.

2

Theory

ML is a subfield of artificial intelligence that has gained significant traction in recent years as it enables models to learn from data and make predictions without being explicitly programmed in a rule-based manner. The field has evolved rapidly, and it encompasses a wide range of techniques, including supervised learning, unsupervised learning, and reinforcement learning. The process typically involves training a model on a dataset to identify patterns and relationships between data points. Once a model is trained, it can run inference on new data.

Advancements in ML, particularly in Deep Learning (DL), have revolutionized the field, enabling machines to achieve state-of-the-art performance on a wide range of complex tasks such as semantic segmentation [15], image classification [16], and OD [17]. All of these are fundamental tasks in computer vision and enable a vast space of application in various domains. Semantic segmentation pertains to, given for instance an input image, output a class prediction from a set of predefined classes for each pixel. Similarly, image classification predicts one or multiple classes, but for an image as a whole. While this is far from trivial, OD takes it further and resolves spatial predictions for each object of interest on the image plane. As such, DL OD generally relies on more complex architectures and techniques to reach sufficient performance levels. DL OD models have shown great promise in a wide range of applications [17]. For instance, autonomous cars use object detection to identify pedestrians, vehicles, and other objects in their environment and make decisions accordingly.

Despite rapid advancements in the field, OD and DL are still active areas of research, and many challenges remain. One of them being computationally expensive and time-consuming training phases. To address this, researchers have been proposing various ways to accelerate training of DL models. Examples include adaptive learning rate [18], momentum [19], normalization [20], weight decay [21], and numerous other variance reduction techniques. While these are all fundamental aspects of accelerating model training, prioritizing informative samples have shown to induce a similar effect [6], [7], [10], [22]. The common trajectory of the performance curve when training deep learning models generally starts with a high rate of convergence, and any clean sample from the dataset may sufficiently contribute when updating the parameters at this stage. As the model continues to learn, the rate of convergence slows down, inferring that a good amount of samples might have been learned and have become too easy for the model to predict correctly. In these instances,

the model is subject to negligible contributions. Intuitively, treating each entry in the dataset equally could be sub-optimal and computing resources could instead be concentrated on informative samples in an effort to accelerate training. In fact, this is a problem well suited for a method titled Importance Sampling (IS) [23], [24].

In this section, we tread through some of the fundamental concepts in deep neural networks in order to later motivate why IS can accelerate training of deep learning models. We will also examine how DL OD works and visit some ideas that have enabled high-performing models in a real-time setting, such as AD. Lastly, we will touch on what importance sampling is, how it works, and its application in deep learning training.

2.1 Multilayer Perceptron

Multilayer Perceptrons (MLPs) are a class of models within ML that are inspired by the structure and function of biological neurons in the brain [25]. These types of models consist of layers of interconnected processing units, also called neurons, that transform an input and propagate the information through the layers until a final output is obtained. This structure has enabled remarkable success in solving highly complex non-linear problems. Before proceeding with the architecture, it is important to understand the computations around a neuron which is visualized as a graph in figure 2.1. The basic computation for a single neuron is expressed as $a = \sigma(\sum_i^n w_i x_i)$ where w_i is a weight scalar, x_i is an element in an input vector $\boldsymbol{x}$ and $\sigma(x)$ is an activation function, which is a crucial part for inducing nonlinear capabilities. Without non-linearity, the output would just be a linear transformation of the input, which limits the network to model merely simple relationships between them.

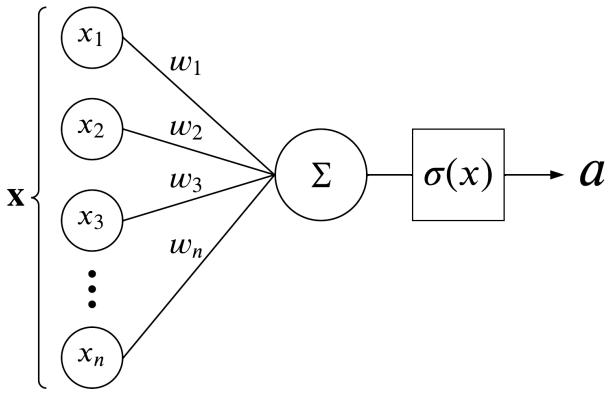

Figure 2.1: A simple computational graph of a neuron

The structure of an MLP is visualized in figure 2.2. It consists of an input layer (the first aggregate of neurons), the output layer (the last aggregate of neurons, 1 in this case), hidden layers which correspond to all the layers between the input and the output layer, and weights visualized as connections between neurons. If we

further vectorize the neuron computation from figure 2.1 to handle an aggregate of neurons we get $\boldsymbol{a} = \sigma(\boldsymbol{W}^{(l)}\boldsymbol{x}^{(l)})$, which corresponds to the vectorized activation from layer l. It is also common but not mandatory to add a bias term $\boldsymbol{b}$ which is simply added to the preactivation output i.e. $\boldsymbol{a} = \sigma(\boldsymbol{W}^{(l)}\boldsymbol{x}^{(l)} + \boldsymbol{b}^{(l)})$. From this, we can express the nested computations which capture the whole forward pass from input to output $\boldsymbol{a}^{(l)} = \sigma(\boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)})$. Fundamentally, the current layer's input is the previous layer's activation output.

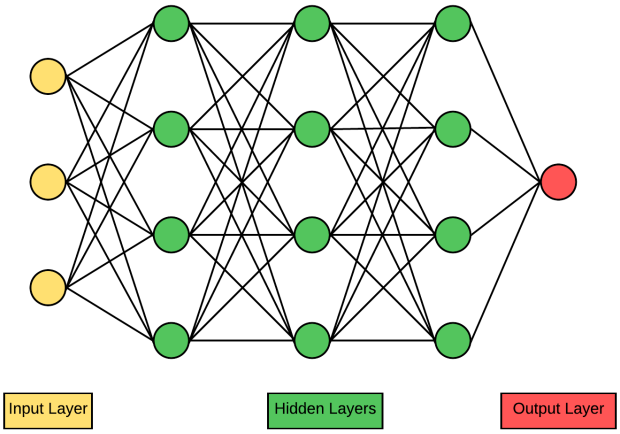

Figure 2.2: The structure of an MLP.

2.1.1 Training a Neural Network

As with many other ML models, when training a neural network, we are trying to optimize an objective function. Informally we want to tweak the parameters $\boldsymbol{\theta}$, i.e. the weights and biases, to accurately map the input to the desired output. The underlying algorithm for achieving this is called Gradient Descent [26]. It works by iteratively stepping in the negative gradient's direction and updating the parameters for the function of interest. This process is visualized in figure 2.3.

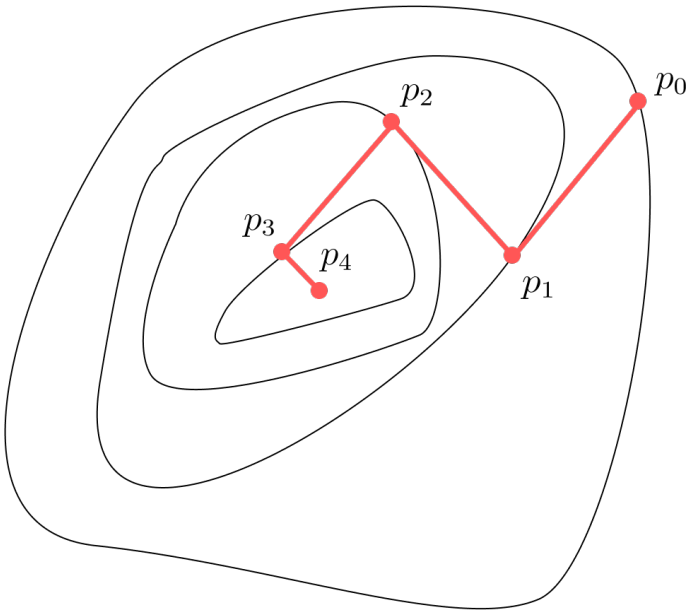

Figure 2.3: A simple case of gradient descent where $\boldsymbol{p_i} \in \mathbb{R}^2$

The gradient $\boldsymbol{\nabla f}$ is a vector-valued function whose components are the partial derivatives of f i. e.

$$\boldsymbol{\nabla f}(\boldsymbol{p}) = \begin{bmatrix} \frac{\partial f(\boldsymbol{p})}{\partial \boldsymbol{p}_1} \\ \vdots \\ \frac{\partial f(\boldsymbol{p})}{\partial \boldsymbol{p}_n} \end{bmatrix}$$

A fundamental property of the gradient is that given a point $\bar{\boldsymbol{p}}$, it points in the direction in which the function locally increases most, and so if we are minimizing, it is in our interest to step in the negative direction where the function decreases the most. The algorithm is outlined below.

Algorithm 1 General gradient descent

Initialize starting point $\boldsymbol{p}$, function f, convergence criterion ϵ
while $||\boldsymbol{\nabla f}(\boldsymbol{p})|| > \epsilon$ **do**
  Find search direction (negative gradient): $\boldsymbol{s} = -\boldsymbol{\nabla f}(\boldsymbol{p})$
  Obtain step size with line search: $\alpha = \arg\min_\alpha f(\boldsymbol{p} + \alpha \boldsymbol{s})$
  Update state: $\boldsymbol{p} \longleftarrow \boldsymbol{p} + \alpha \boldsymbol{s}$
end while

In the context of neural networks, $\bar{\boldsymbol{p}}$ is a point in space of possible parameters $\bar{\boldsymbol{\theta}}$. The objective function we want to minimize is often referred to as a loss function (also known as a cost function). Given an MLP parameterized with $\boldsymbol{\theta}$ the loss function $L_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})$ quantifies how poor the prediction was according to a ground truth $\boldsymbol{y}$. Let $J(\boldsymbol{\theta}) = \mathbb{E}_\pi[L_{\boldsymbol{\theta}}(\boldsymbol{x}, \mathbf{y})], (\boldsymbol{x}, \boldsymbol{y}) \sim \pi$. Essentially we want to obtain the optimal parameters that yield the lowest expected loss for our problem at hand: $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$.

Before we can apply the concept of Gradient Descent, we must be able to compute the gradient of the loss function. The gradient will essentially inform the model in what direction the parameters $\boldsymbol{W}^{(l)}$ and $\boldsymbol{b}^{(l)}$ should be updated in order to minimize the loss, which consequently improves the performance of the model. An algorithm for computing the gradients is referred to as Backpropagation [27]. The idea of Backpropagation is to propagate the loss back through the network, from the output layer to the input layer, using the chain rule of calculus. Let us assume that we run a forward pass and have computed the loss L. Backpropagation computes all the partial derivatives of the loss with respect to each weight ($\frac{\partial L}{\partial \boldsymbol{W}_{ij}^{(l)}}$) and with respect to each bias ($\frac{\partial L}{\partial \boldsymbol{b}_{j}^{(l)}}$) in order to obtain the components of the gradient. These partial derivatives hold information about the change in loss w.r.t. to the change in parameters.

Now consider an example where we have one output layer and two hidden layers i.e. $\hat{\boldsymbol{y}} = \sigma(\boldsymbol{W}^{(3)}\sigma(\boldsymbol{W}^{(2)}\sigma(\boldsymbol{W}^{(1)}\boldsymbol{x})))$ and $\boldsymbol{a}^{(l)} = \sigma(\boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)})$. Note that we have omitted the bias term to simplify the example. Let all the activation functions denote the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$, and let the non-parametrized loss function denote a simple mean squared error function $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{2}(\boldsymbol{y}-\hat{\boldsymbol{y}})^2$, which can also be expressed explicitly as a function of all the weights of each layer:

$$\begin{aligned} L(\hat{\boldsymbol{y}}, \boldsymbol{y}) &= L(\sigma(\boldsymbol{W}^{(3)}\boldsymbol{a}^{(2)}), \boldsymbol{y}) \\ &= L(\sigma(\boldsymbol{W}^{(3)}\sigma(\boldsymbol{W}^{(2)}\boldsymbol{a}^{(1)})), \boldsymbol{y}) \\ &= L(\sigma(\boldsymbol{W}^{(3)}\sigma(\boldsymbol{W}^{(2)}\sigma(\boldsymbol{W}^{(1)}\boldsymbol{x}))), \boldsymbol{y}) \end{aligned}$$

We have the following derivatives for the non-parameterized loss function and the sigmoid activation function:

$$\begin{aligned} \frac{\partial L}{\partial \hat{\boldsymbol{y}}} &= \hat{\boldsymbol{y}} - \boldsymbol{y} \\ \frac{\partial \sigma}{\partial x} &= \sigma(x)(1 - \sigma(x)) \end{aligned}$$

Now to obtain the partial derivatives we work backward through the network to calculate $\frac{\partial L}{\partial \boldsymbol{W}^{(3)}}$ then $\frac{\partial L}{\partial \boldsymbol{W}^{(2)}}$ and then $\frac{\partial L}{\partial \boldsymbol{W}^{(1)}}$. The calculation for the gradient in the third layer gives

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{W}^{(3)}} &= \frac{\partial L}{\partial \hat{\boldsymbol{y}}} \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{W}^{(3)}} \\ &= (\hat{\boldsymbol{y}} - \boldsymbol{y})\sigma(\boldsymbol{W}^{(3)}\boldsymbol{a}^{(2)})\sigma(1 - (\boldsymbol{W}^{(3)}\boldsymbol{a}^{(2)}))\boldsymbol{a}^{(2)} \end{aligned}$$

For calculating the gradient in the second layer, we have to use the chain rule twice:

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{W}^{(2)}} &= \frac{\partial L}{\partial \hat{\boldsymbol{y}}}\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}^{(2)}}\frac{\partial \boldsymbol{a}^{(2)}}{\partial \boldsymbol{W}^{(2)}} \\ &= \underbrace{(\hat{\boldsymbol{y}} - \boldsymbol{y})}_{\frac{\partial L}{\partial \hat{\boldsymbol{y}}}} \underbrace{\sigma(\boldsymbol{W}^{(3)}\boldsymbol{a}^{(2)})\sigma(1 - (\boldsymbol{W}^{(3)}\boldsymbol{a}^{(2)}))\boldsymbol{W}^{(3)}}_{\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}^{(2)}}} \underbrace{\sigma(\boldsymbol{W}^{(2)}\boldsymbol{a}^{(1)})(1 - \sigma(\boldsymbol{W}^{(2)}\boldsymbol{a}^{(1)}))\boldsymbol{a}^{(1)}}_{\frac{\partial \boldsymbol{a}^{(2)}}{\partial \boldsymbol{W}^{(2)}}}\end{aligned}$$

Similarly, the gradient for the first layer can be calculated from the following expression

$$\frac{\partial L}{\partial \boldsymbol{W}^{(1)}} = \frac{\partial L}{\partial \hat{\boldsymbol{y}}}\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}^{(2)}}\frac{\partial \boldsymbol{a}^{(2)}}{\partial \boldsymbol{a}^{(1)}}\frac{\partial \boldsymbol{a}^{(1)}}{\partial \boldsymbol{W}^{(1)}}$$

.

Note that some factors appear in multiple calculations, and so computationally, it is not necessary to re-calculate these expressions. Only the factors that are particular to the layer must be evaluated while the factors that are common to previous layers can easily be reused.

Since we now have a way of deriving the gradient, we can similarly to algorithm 1 outline gradient descent in the context of training a neural network:

Algorithm 2 Gradient descent for training a neural network

Input initial parameters $\boldsymbol{\theta}$, learning rate η, data $D = \{(\boldsymbol{x_i}, \boldsymbol{y_i})\}_{i=0}^{N}$
for t in $1, 2, 3, ..., T$ **do**
  Randomly sample $D_t \subseteq D$ where $|D_t| = n_b$ is the batch size
  $\boldsymbol{x}, \boldsymbol{y} \longleftarrow D_t$
  Compute output from forward pass: $\hat{\boldsymbol{y}} \longleftarrow Forward(\boldsymbol{x})$
  Compute loss: $L \longleftarrow L(\hat{\boldsymbol{y}}, \boldsymbol{y})$
  Compute average gradient across the batch: $\hat{\boldsymbol{g}} \longleftarrow \frac{1}{n_b}\sum_{i=0}^{n_b} \boldsymbol{\nabla} \boldsymbol{L}_{\boldsymbol{\theta}}^{(i)}$ from $\boldsymbol{\nabla} \boldsymbol{L}_{\boldsymbol{\theta}}^{(i)} \longleftarrow Backward(L)$
  Update parameters $\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} - \eta\hat{\boldsymbol{g}}$
end for

The following clarifications should be noted

- The above process is usually repeated for a couple of so-called epochs.
- The learning rate η is a hyperparameter influencing the magnitude of the update step.
- D_t usually selects a subset of samples from D by sampling uniformly without replacement.

As inferred by the last point, since gradient descent is an iterative algorithm, when applying it to train a neural network, the process must define the quantity of training data to propagate at each iteration. In theory, it is possible to forward- and

backpropagate on the whole dataset each step i.e. $n_b = |D|$. In fact, assuming that the training data is a perfect representation of the real-world problem, this would yield in the most optimal direction to step in. However, the computation and memory consumption required to perform this operation makes this infeasible in practice for large-scale datasets. This approach is known as batch gradient descent. On the other extreme, we define stochastic gradient descent. Instead, of propagating the whole dataset each iteration, this approach selects only one training example per iteration i.e. $n_b = 1$. The number of iterations per time unit would be remarkably better and the memory consumption would be kept at a minimum. However, since the whole parameter space is influenced by solely one training example per iteration, this approach suffers from highly fluctuating gradients and is very prone to stepping in sub-optimal directions. As a compromise, mini-batch gradient descent propagates multiple examples per step. The number of samples per batch is typically determined by a hyperparameter which is set to accommodate hardware capabilities. This approach is less prone to high variance than stochastic gradient descent, and is computationally feasible, unlike batch gradient descent in most cases.

2.1.2 Convolutional Neural Networks

MLPs are generally not the preferred way to solve computer vision tasks. If the input/output has to deal with large images, the model will generally suffer from heavy parameterization as each layer is fully connected. Furthermore, since the input and the weights have a static layout where each pixel will be mapped to one input node, the model will not be spatially invariant which is could be a desirable property of models in computer vision. Spatial invariance allows the model to capture features regardless of their position in the input image [28].

Convolutional Neural Network (CNN) is a type of neural network that is commonly applied in computer vision tasks. At the core of a CNN is the convolution which is an operation that allows the network to extract useful features from images while being spatially invariant [28]. It works by applying a set of learnable filters, one small region at a time, and produces a new feature map as illustrated in figure 2.4. Specifically, one convolution takes the dot product between the current region of the image across channels and the filter to obtain a preactivation scalar. The value of the preactivation denotes how well the convolved area matches the values/pattern in the filter. The filters slide over the image, performing a Hadamard product followed by a sum of the elementwise products at each position which in the end forms a complete feature map. By nesting this operation over a stack of layers with multiple filters per layer, CNNs are able to learn increasingly complex and hierarchical features from the input. Much like in MLPs, the feature maps from each layer are passed through an activation function to induce non-linearity. The hyperparameters for a convolutional layer are primarily:

- kernel size - The size of the weight matrix that corresponds to a filter
- stride - Defines the step size of the filter as it slides over the input
- padding - An offset that defines how far out from the input's border the filter

can go

- number of channels - Number of filters in the layer. This number also denotes the number of feature maps the output will have.

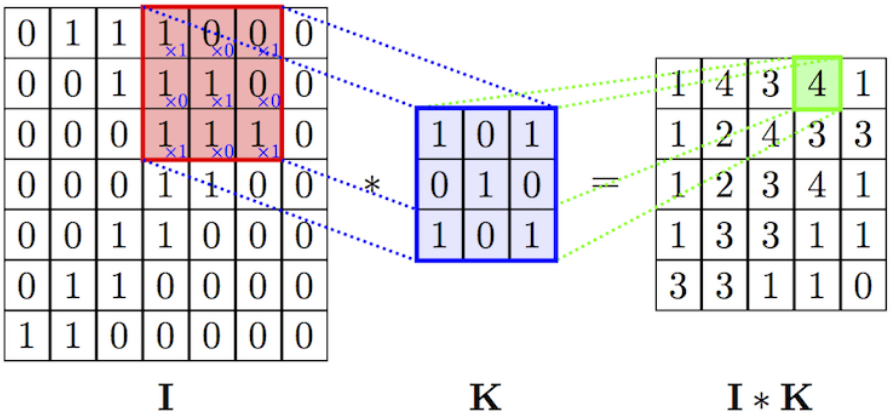

Figure 2.4: A convolution operation [1]. $\boldsymbol{I}$ is the input, $\boldsymbol{K}$ represents the filter and $\boldsymbol{I} * \boldsymbol{K}$ is the resulting feature map.

2.2 Object Detection

OD is a complex task in computer vision that pertains to identifying and localizing objects of interest in an image. Not only does the algorithm have to be able to classify which types of objects are depicted, but it also needs the ability to propose assigned regions for each identified object on the image plane. These region proposals are oftentimes represented by either 2- or 3-dimensional bounding boxes, and when visualized, overlayed on the image as illustrated in figure 2.5. OD has seen numerous applications in various fields such as robotics, healthcare, surveillance, and AD. It is typically used as a critical subsystem to solve many different problems [17], [29]. An example is AD where one or more cameras are mounted to the car and the car has to be able to decipher its surrounding in order to act in a highly dynamic environment.

[1] Detection and Tracking of Pallets using a Laser Rangefinder and Machine Learning Techniques - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/An-example-of-convolution-operation-in-2D-2_fig3_324165524 [accessed 20 Mar, 2023]

Figure 2.5: A training example from Zenseact Open Dataset (ZOD)[2][30] where the 2d bounding box annotation has been overlayed onto the image.

Due to the task's complex intrinsic nature, the state-of-the-art has mainly been dominated by DL approaches in recent years [17]. These methods have rapidly revolutionized the field using CNNs as a core component. For this section, we will present some project-relevant concepts of DL OD such as Single-Stage Detector (SSD), anchor-free detectors, and Non-Maximum Suppression (NMS), along with a thorough walk-through of a specific well-established object detector which applies all the aforementioned concepts. However, before we jump into any details, we will describe the theory on how to evaluate object detectors to get a sense of what the end goal is regardless of how the optimization problem is formulated.

2.2.1 Evaluation Metrics

Evaluation is a crucial part of ML. Rigorously defined and tested evaluation metrics is the standard way of measuring the performance of ML models. There are several evaluation metrics that are commonly used in OD, including Intersection over Union (IoU), Precision and Recall, F1 score, Average Precision (AP), Mean Average Precision (mAP), etc. This section intends to describe the metrics used in the thesis in more detail.

2.2.1.1 Intersection over Union

The Intersection over Union (IoU) of two geometric shapes S_1 and S_2 measures to what extent the two shapes overlap. This is used extensively in OD to determine the overlap of bounding boxes, as a high overlap between predicted and ground truth boxes indicates good performance for the model. The IoU ranges from 0 to 1, where a value of 0 indicates no overlap between the bounding boxes, and a value of 1 indicates perfect overlap. For 2D OD, let $S_1 = \boldsymbol{b_1}$ be a bounding box represented by the coordinate tuple $\boldsymbol{b_1} = (x_1^{(1)}, y_1^{(1)}, x_2^{(1)}, y_2^{(1)})$ where $(x_1^{(1)}, y_1^{(1)})$ is the top-left

[2]For this dataset, Zenseact AB has taken all reasonable measures to remove all personally identifiable information, including faces and license plates. To the extent that you like to request removal of specific images from the dataset, please contact privacy@zenseact.com.

coordinate and $(x_2^{(1)}, y_2^{(1)})$ the bottom-right coordinate, and let $S_2 = \boldsymbol{b_2}$ follow a similar definition. A visual representation of this can be seen in figure 2.6. The area of the intersection between the bounding boxes can be calculated as follows

$$\begin{aligned} |\boldsymbol{b_1} \cap \boldsymbol{b_2}| =& max(0, min(x_2^{(2)}, x_2^{(2)}) - max(x_1^{(1)}, x_1^{(2)})) \\ &\cdot max(0, min(y_2^{(1)}, y_2^{(2)}) - max(y_1^{(1)}, y_1^{(2)})) \end{aligned} \tag{2.1}$$

The area of the union is simply the summed area of the bounding boxes minus the area of the intersection, i.e.

$$\begin{aligned} |\boldsymbol{b_1} \cup \boldsymbol{b_2}| =& (x_2^{(1)} - x_1^{(1)}) \cdot (y_2^{(1)} - y_1^{(1)}) + \\ & (x_2^{(2)} - x_1^{(2)}) \cdot (y_2^{(2)} - y_1^{(2)}) - |\boldsymbol{b_1} \cap \boldsymbol{b_2}| \end{aligned} \tag{2.2}$$

The IoU of the two bounding boxes is simply

$$IoU(\boldsymbol{b_1}, \boldsymbol{b_2}) = \frac{|\boldsymbol{b_1} \cap \boldsymbol{b_2}|}{|\boldsymbol{b_1} \cup \boldsymbol{b_2}|} \tag{2.3}$$

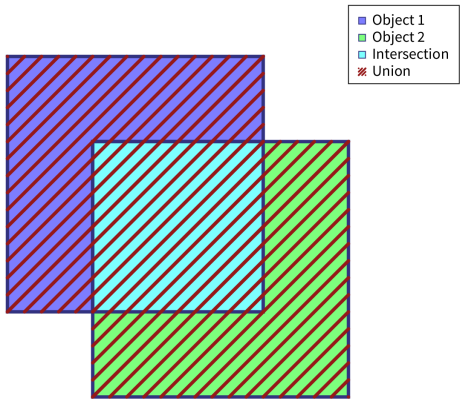

Figure 2.6: A visual representation of the components of the IoU between two 2D bounding boxes.

2.2.1.2 Confusion Matrix

The confusion matrix is a set of valuable metrics that measure how many classifications were correct and how many were wrong. In the case of binary classification, it is a 2-by-2 matrix where the rows and columns represent the ground truth and

predictions respectively, see figure 2.7. One row and one column represent positive locations, and one of each represents negative locations. The result is a matrix where we can easily read how many correct and incorrect predictions the model made of each type, i.e. the true and false positives and negatives. They work as follows:

- True Positives (TPs): The number of samples that were supposed to be true, and that the model predicted to be true.
- True Negatives (TNs): The number of samples that were supposed to be false, and that the model predicted to be false.
- False Positives (FPs): The number of samples that were supposed to be false, but that the model predicted to be true.
- False Negatives (FNs): The number of samples that were supposed to be true, but that the model predicted to be false.

In the case of multi-class classification with C classes, the matrix is C-by-C. In this case, each row and each column represent a class. The matrix value at row i and column j then represents the number of times the model predicted class j when the ground truth class was i.

In OD, TPs are counted every time there is a ground truth object and a predicted object with the same class and an IoU above some threshold. FPs are counted when there is a predicted object which does not overlap sufficiently with any ground truth objects of the same class. FNs are counted when there is a ground truth object which does not overlap sufficiently with any predicted object of the same class.

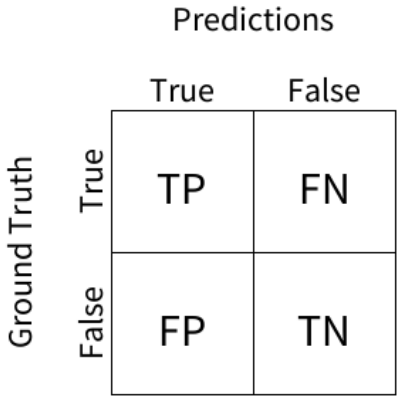

Figure 2.7: A binary confusion matrix

2.2.1.3 Average Precision

AP is the average precision over the recall-values [31]. Precision is the fraction of correct positive classifications out of all the positive classifications:

$$P = \frac{TP}{TP + FP}$$

Recall is the fraction of correct positive classifications out of all the actual positive cases:

$$R = \frac{TP}{TP + FN}$$

Preferably, we want both precision and recall to be high (close to 1), but there is typically a trade-off between them for a given model; as the recall increases, the precision tends to decrease. To compute AP, we first rank all the object predictions by their confidence score in descending order. Then, going from top to bottom, computing and storing the current precision and recall value for each time we visit a new prediction yields a sequence of precision-recall pairs which we can use to compute the final metric. If we plot this, we get the precision-recall curve, as in figure 2.8. As can be seen in the figure, the precision is wiggling up and down in a sort of zigzag pattern as the recall increases, which is due to the variations of correct and incorrect predictions when iterating through the ranked detections [32].

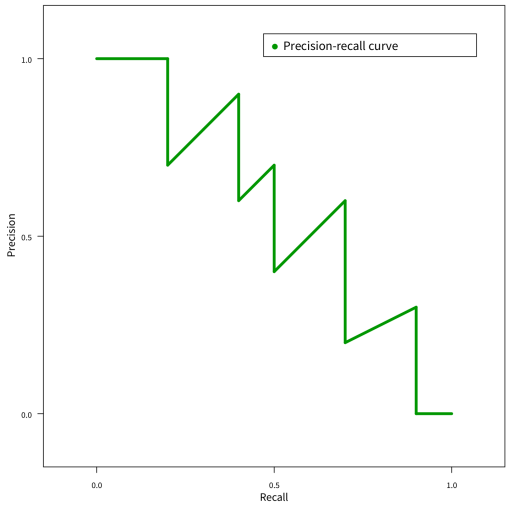

Figure 2.8: An example precision-recall curve.

In order to account for the variations, and as a means to increase the robustness of the metric, it is standard to, for each point i, define the interpolated precision-value P_i^{interp} to be the maximum precision-value of all recall-values at or above i: $P_i^{interp} = max(P_i, P_{i+1}, ..., P_n)$. The AP is now the average interpolated precision per recall value, which is equivalent to the area under the interpolated precision-recall curve, see figure 2.9.

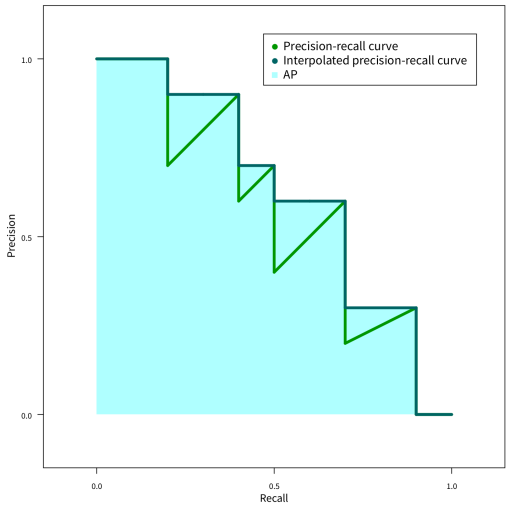

Figure 2.9: An example precision-recall curve with and without interpolated precision-values.

Note that, in the literature, several versions of AP are used, based mainly on the dataset that the model is being evaluated on. For example, the PASCAL-VOC Challenge [32] used to compute the AP of each class using 11 equally spaced recall values, but was changed in 2010 to use all values.

2.2.1.4 Mean Average Precision

mAP is the standard way of measuring the performance of an OD model. The AP is computed separately for each class, and the mAP is the mean of the class-wise APs. Since the expectation value of AP is invariant to the number of objects, mAP is an unbiased estimation of the true mAP given an infinite dataset. Note that the literature sometimes uses the terms AP and mAP interchangeably.

2.2.2 Single-stage Object Detectors

Overall, DL OD architectures can be divided into two categories: two-stage object detectors and SSDs. As the name suggests, two-stage object detectors consist of two stages i.e. region proposal followed by object classification. In the first stage, the algorithm generates a set of candidate object locations and in the second stage, it classifies each region proposal into one of the object categories or background. Two-stage detectors are known for their high performance but are generally slow and computationally inefficient as a result of the separate handling of tasks. In contrast, SSDs deal with both sub-problems in a single forward pass, hence enabling efficiency, speed, and simplicity. Nonetheless, this generally results in worse performance when compared to their two-staged counterpart. While high performance is crucial in the context of AD, two-stage detectors are commonly deemed obsolete as the algorithms must, by requirement, be able to operate in real-time. With that being said, this thesis adopts an SSD approach.

2.2.3 Anchor-free Detectors

In traditional DL OD, anchor boxes are used to generate object proposals and refine object localization. They are in essence a set of predefined fixed-sized rectangles of various scales and aspect ratios evenly distributed across the image. During training, the model is trained to predict the offsets and scales between each relevant anchor box and the corresponding ground truth bounding box for each object in an image. This allows the model to better match the object's bounding box in the image in comparison to a naive approach where the predicted bounding box is limited to a single aspect ratio and/or scale. However, anchor boxes are also limited in a sense as they have to be predefined and adjusted specifically for the dataset and domain at hand. If the predefined anchor boxes do not align well with the bounding boxes in the dataset, the model might not be able to accurately localize the objects.

Anchor-free object detectors are a type of OD models that do not rely on predefined templates in order to localize objects in an image. Instead, they typically predict objects' locations by directly predicting all the necessary bounding box parameters. These are obtained from extracted features from the output of deep convolutional layers. Anchor-free models tend to have a host of benefits as they are simple, flexible, and agnostic to the distribution of bounding box scales and aspect ratios for any given dataset.

Regarding anchor-free detectors, it is worth mentioning that they are commonly presented in the context of dense-based object detectors. These object detectors project their predictions onto a dense prediction map which is a high dimensional array whose shape resembles the shape of the input image. As such, they enable fine-grained localization and instance segmentation that offers rich and detailed information about the input. In contrast to these types of object detectors are models that view the problem as a set prediction problem. The output from a set-based object detector for a given sample is simply a set of class and bounding box tuples. Transformer models are notably good at this, since they work by taking a set as input to output a set of features [33]. A well-known set-based framework is DEtection TRansformer (DETR) where image features are extracted and divided and spatially embedded into a set of object queries [34]. The object queries serve as inputs to a transformer model which is followed by a prediction head to output a set of predictions. While set-based object detectors can omit certain post-processing techniques, the computations revolving around the transformer module can be costly when there are a lot of object queries to be processed. Set-based object detectors are also arguably less interpretable since they do not offer any explicit spatial references in the predictions as opposed to a dense prediction map. For these reasons, this thesis employs a dense-based object detector which is explained in greater detail in 4.1.

2.2.4 Non-Maximum Suppression

Dense-based OD models typically generate multiple candidate bounding boxes around objects of interest, leading to the need to filter out some of these boxes in order to keep just one bounding box per relevant object. NMS is a post-processing technique

used to filter redundant overlapping object region proposals. It does so by identifying and removing proposals with an IoU greater than a predefined iou-threshold when comparing to the proposal with the highest confidence score for the same class (see figure 2.10). It is also common to filter out bounding boxes with a confidence score below a predefined confidence threshold beforehand. The overarching procedure is expressed in the following pseudo code:

Algorithm 3 Pseudo code of NMS

Input: $B = \{\boldsymbol{b_1}, ..., \boldsymbol{b_N}\}, S = \{s_1, ..., s_N\}, th_{iou}, th_{conf}$
B: collection of bounding boxes
S: collection of corresponding confidence scores
th_{iou}: IoU threshold
$th_{confidence}$: confidence threshold

for i in $0, ..., N$ **do**
  if $s_i < th_{conf}$ **then**
    $B \leftarrow B \setminus \boldsymbol{b_i}$
    $S \leftarrow S \setminus s_i$
  end if
end for

$\hat{B} \leftarrow \emptyset$
while $B \neq \emptyset$ **do**
  $m \leftarrow argmax(S)$
  $\hat{B} \leftarrow \hat{B} \cup \boldsymbol{b_m}$
  $B \leftarrow B \setminus \boldsymbol{b_m}$
  for j in 0, ..., $|B|$ **do**
    if $iou(\boldsymbol{b_m}, \boldsymbol{b_j}) \geq th_{iou}$ **then**
      $B \leftarrow B \setminus \boldsymbol{b_j}$
      $S \leftarrow S \setminus s_j$
    end if
  end for
end while
return $\hat{B}, S$

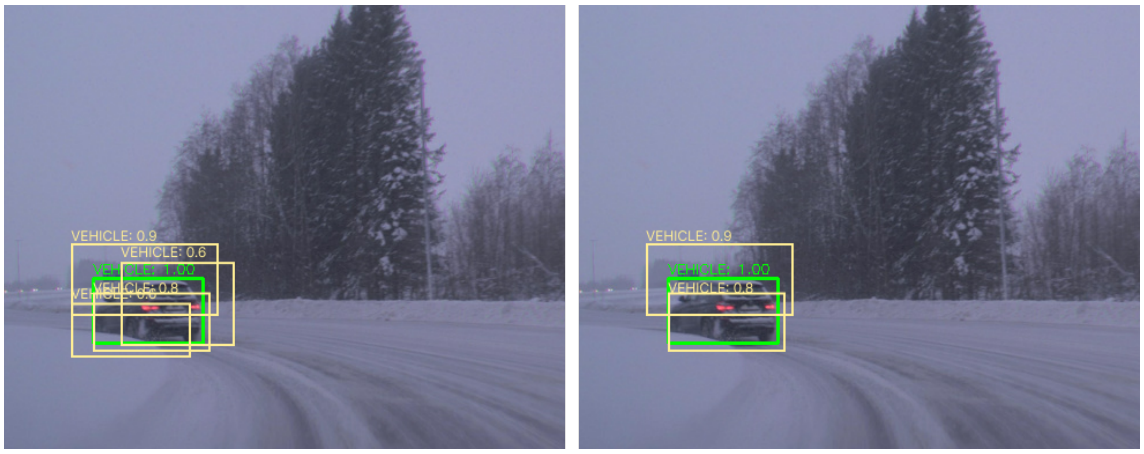

Figure 2.10: An example of non-maximum suppression. The left image depicts the candidate bounding box predictions before non-maximum suppression. The right image depicts the processed image, where two bounding box predictions were filtered out as they overshot the iou-threshold with another more confident prediction.

2.2.5 Fully Convolutional One-Stage Object Detection

The object detector that was used for this thesis resembles but is not exactly equal to, a model called Fully Convolutional One-Stage Object Detector (FCOS) [35]. It is an anchor-free single-stage object detector, where the main idea is to perform object detection directly on feature maps produced by a Fully Convolutional Network (FCN). It does so by projecting its class and bounding box predictions on a dense prediction map with a shape similar to the input image, but downscaled and with more channels. As such, the predictions can be interpreted in a per-pixel manner which somewhat resembles how a typical DL image semantic segmentation network works [15]. For each target pixel i.e. a strided region of the input image, the model predicts a class score and a left, top, right bottom offset from the location (x, y). The left, top right, and bottom predictions form a box where the pixel's location is an interior point and the offsets define the distances from the interior point to the corresponding box boundaries. Let the ground-truth objects for an input image be defined as $\boldsymbol{o_i}$ where $\boldsymbol{o_i} = (c^{(i)}, x_1^{(i)}, y_1^{(i)}, x_2^{(i)}, y_2^{(i)}) \in \{1, 2, ..., C\} \times \mathbb{R}^4$. Here $(x_1^{(i)}, y_1^{(i)})$, $(x_2^{(i)}, y_2^{(i)})$ is the left-top and right-bottom coordinate of object i's bounding box respectively. $c^{(i)}$ is the class index where C is the number of classes. For a location (x, y) in the original image, there exists a corresponding target location $(\frac{s}{2}+xs, \frac{s}{2}+ys)$ on the s strided target feature map. Each location can be seen as a separate pixel sample, and a location is considered positive if it lies within a ground truth bounding box; otherwise negative. Naturally, each location holds the ground truth class c^* and a regression target $\boldsymbol{u}^* = (l^*, t^*, r^*, b^*)$, where the elements are the ground truth offset in the left, top, right and bottom direction respectively. Assuming location

(x, y) is a positive location for ground truth $\boldsymbol{o_i}$, the regression target becomes:

$$\begin{aligned} l^*_{x,y} &= \frac{(x - x_1^{(i)})}{s} \quad t^*_{x,y} = \frac{(y - y_1^{(i)})}{s} \\ r^*_{x,y} &= \frac{(x_2^{(i)} - x)}{s} \quad b^*_{x,y} = \frac{(y_2^{(i)} - y)}{s} \end{aligned} \tag{2.4}$$

If a location falls within two different objects' bounding boxes, the location is denoted as an ambiguous sample, and the smallest object takes priority when constructing the target feature map.

2.2.5.1 Architecture

The network architecture proposed by Tian *et al.* [35] is an Feature Pyramid Network (FPN) layered on top of a ResNet backbone. An FPN acts as a modular component in a CNN that takes a feature map and generates a proportionally sized feature map at multiple levels. One benefit of this is to alleviate the problem of objects' scale invariance. The higher resolution feature maps are more detail-aware, whereas the lower resolution feature maps are more context-aware. Fusing happens between adjacent layers to make benefit from context-aware features. Furthermore, each level in the FPN tunnels to a shared head which outputs a dense prediction map. The benefit of multi-level predictions is twofold. Firstly, the difficulty of identifying objects in a large span of different sizes is alleviated. Secondly, ambiguous samples can be stratified and impose less of a problem. For this to work properly each output must have a corresponding target feature map where the objects are stratified by size. All targets are initialized the same, but then for each target pixel of each feature level i, a location is converted to a negative location if it satisfies $max(l^*_{x,y}, t^*_{x,y}, r^*_{x,y}, b^*_{x,y}) > m_i$. m_i is simply a predefined threshold of the max distance from the location center and upper bounds the maximum size of objects for a given feature level.

For each location in the dense prediction map, the network outputs a prediction for the 4 regression targets i.e. $\boldsymbol{u}_{x,y} = (l_{x,y}, t_{x,y}, r_{x,y}, b_{x,y})$, and a class prediction vector $\boldsymbol{c}_{x,y}$. These are stacked along the channel dimension, and the full dense prediction will be $\boldsymbol{y} \in \mathbb{R}^{\frac{h}{s} \times \frac{w}{s} \times C \times 4}$. As such, for each target location, C binary classifiers are trained for the classification task along with 4 regressors. Since the model is trained to perform both classification and bounding box regression, the final activation functions are separated for each task. Sigmoid $\sigma(\boldsymbol{c}_{x,y,z}) = \frac{1}{1+e^{-\boldsymbol{c}_{x,y,z}}}$ is used for each (binary) classification prediction and rectified linear unit $ReLu(\boldsymbol{a}) = max(\boldsymbol{0}, \boldsymbol{a})$ is used for the regression predictions.

2.2.5.2 Loss Function

The loss function has two terms due to the duality of the problem and is defined as follows:

$$\begin{aligned} L((\boldsymbol{c}_{x,y}, \boldsymbol{u}_{x,y}), (\boldsymbol{c^*}_{x,y}, \boldsymbol{u^*}_{x,y})) = &\frac{1}{N_{pos}} \sum_{x,y} L_{cls}(\boldsymbol{c}_{x,y}, \boldsymbol{c^*}_{x,y}) \\ &+ \frac{\lambda}{N_{pos}} \sum_{x,y} \mathbb{1}_{\boldsymbol{c^*}_{x,y} > 0} L_{reg}(\boldsymbol{u}_{x,y}, \boldsymbol{u}^*_{x,y}) \end{aligned} \tag{2.5}$$

Here L_{reg} is the GIoU loss i.e. $L_{reg} = 1 - (\frac{|\boldsymbol{b} \cap \boldsymbol{b}^*|}{|\boldsymbol{b} \cup \boldsymbol{b}^*|} - \frac{|\boldsymbol{h} \setminus (\boldsymbol{b} \cup \boldsymbol{b}^*)|}{|\boldsymbol{h}|})$, which describes an inverse metric of the overlap between the predicted bounding box $\boldsymbol{b}$ derived from $\boldsymbol{u}_{x,y}$, the ground truth bounding box $\boldsymbol{b}^*$ derived from $\boldsymbol{u}^*_{x,y}$ and the smallest box $\boldsymbol{h}$ that encloses both $\boldsymbol{b}$ and $\boldsymbol{b}^*$. The first term in the parenthesis is equal to the IoU, an overlap metric that is described in more detail in 2.2.1.1. Furthermore, $\mathbb{1}_{\boldsymbol{c^*}_{x,y}>0}$ is an indicator function which is 1 if the current location is a positive location. N_{pos} is simply the number of positive locations. L_{cls} is the binary focal loss computed over each entry in the predicted probability vector $\boldsymbol{c}_{x,y} \in \mathbb{R}^C$ against the ground truth class $c^*_{x,y}$. Focal loss is a modification of the regular cross-entropy loss and can be expressed as follows

$$FL(\boldsymbol{c}_{x,y,z}, \boldsymbol{c^*}_{x,y}) = \begin{cases} -\alpha(1 - \boldsymbol{c}_{x,y,z})^\gamma \log(c_{x,y,z}) & \text{if } z = c^*_{x,y} \\ -(1-\alpha)c^\gamma_{x,y,z} \log(1 - c_{x,y,z}) & \text{otherwise} \end{cases} \tag{2.6}$$

α is a balancing factor, and γ is a hyperparameter that reduces the relative loss for well-classified examples. This gives $L_{cls}(\boldsymbol{c}_{x,y}, \boldsymbol{c^*}_{x,y}) = \sum_{i=0}^{C} FL(c_{x,y,i}, \boldsymbol{c^*}_{x,y})$. Finally, λ is just a balancing factor, commonly set to 1. Worth mentioning is that FCOS applies a concept known as centerness to suppress low-quality bounding boxes in the post-processing stage, which adds a new component to the loss and the predictions. This thesis does not employ this concept to keep a cleaner scientific setting in regards to IS and will not go into further detail about it.

2.3 Importance Sampling

Importance Sampling (IS) is commonly known as a Monte Carlo method that plays a crucial role in the estimation of statistical properties of complex systems [24] [23]. It operates on the premise that certain values of the input random variables carry greater significance when estimating a value compared to others. By placing a higher emphasis on these important values through more frequent sampling, the variance of the estimate given these types of samples can be decreased.

Formally, in the continuous case, we are interested in calculating

$$\int p(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x} = \mathbb{E}_p[f(\boldsymbol{x})]$$

where p is the target distribution and $f(\boldsymbol{x})$ being the function of interest. However, if the dimensionality of $\boldsymbol{x}$ is high, the integral may be very difficult to compute. We could possibly approximate this expectation using a numerical approximation method that computes the following average:

$$\mathbb{E}_p[f(\boldsymbol{x})] \approx \underbrace{\frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x_i})}_{\boldsymbol{s}} \quad \boldsymbol{x_i} \sim p$$

Note that since $\boldsymbol{x_i}$ is drawn from a probability distribution, it infers that we could redraw a new sample average $\boldsymbol{s}$, which assumes its own probability distribution. In

fact, as N tends to infinity, by the central limit theorem [36], $\boldsymbol{s}$ will follow a Gaussian distribution centered around the true expectation $\mathbb{E}_p[f(\boldsymbol{x})]$ i.e.

$$\boldsymbol{s} \xrightarrow{dist} \mathcal{N}(\mu, \sigma^2) \begin{cases} \mu = \mathbb{E}_p[f(\boldsymbol{x})] \\ \sigma^2 = \frac{1}{N}\mathrm{Var}_p[f(\boldsymbol{x})] \end{cases}$$

Now, by introducing a proposal distribution q, which should be selected heuristically, we have the possibility to bias the sampling while maintaining the estimand in question. The trick is as follows

$$\begin{aligned} \mathbb{E}_p[f(\boldsymbol{x})] &= \int p(\boldsymbol{x}) f(\boldsymbol{x}) \frac{q(\boldsymbol{x})}{q(\boldsymbol{x})} d\boldsymbol{x} \\ &= \int q(\boldsymbol{x}) [\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} f(\boldsymbol{x})] d\boldsymbol{x} \\ &= \mathbb{E}_q[\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} f(\boldsymbol{x})] \end{aligned}$$

We are yet again subject to approximating this new integral using a numerical approximation method but this time sampling from q:

$$\mathbb{E}_q[\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} f(\boldsymbol{x})] \approx \underbrace{\frac{1}{N} \sum_{i=1}^{N} \frac{p(\boldsymbol{x_i})}{q(\boldsymbol{x_i})} f(\boldsymbol{x_i})}_{\boldsymbol{r}} \quad \boldsymbol{x_i} \sim q$$

As before, we can conclude that the distribution of $\boldsymbol{r}$ will approach a Gaussian distribution with the following parameters when N approaches infinity:

$$\boldsymbol{r} \xrightarrow{dist} \mathcal{N}(\mu, \sigma^2) \begin{cases} \mu = \mathbb{E}_q[\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} f(\boldsymbol{x})] \\ \sigma^2 = \frac{1}{N}\mathrm{Var}_q[\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} f(\boldsymbol{x})] \end{cases}$$

Interestingly, we now have a new, possibly improved variance. The aim is to satisfy $\frac{1}{N}\mathrm{Var}_q[\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} f(\boldsymbol{x})] < \frac{1}{N}\mathrm{Var}_p[f(\boldsymbol{x})]$ as this will effectively decrease the uncertainty in the estimate of $E_p[f(\boldsymbol{x})]$. To do this, $q(\boldsymbol{x})$ should be high where $|p(\boldsymbol{x})f(\boldsymbol{x})|$ is high. With an appropriately chosen q, the sampling effort is concentrated on the regions of which the integrand of the target distribution has high values, leading to more accurate results with fewer samples.

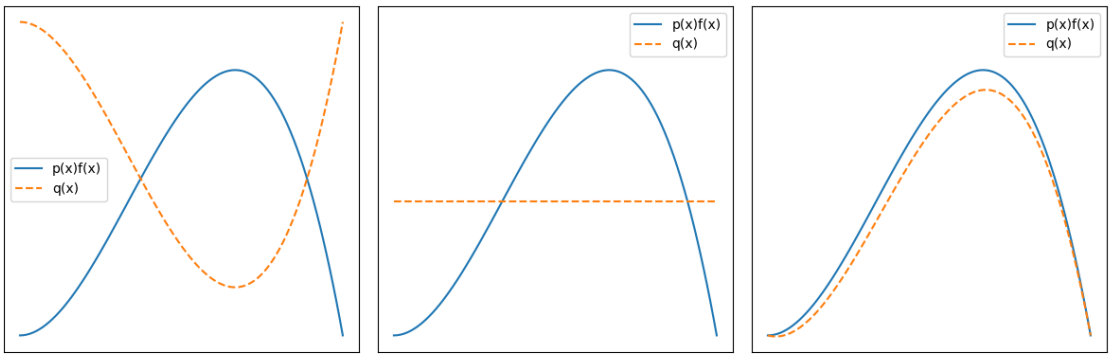

Figure 2.11: Comparison of three different choices for $q(\boldsymbol{x})$. The rightmost plot showcases an example where the choice of $q(\boldsymbol{x})$ would provide a favorable variance reduction over the uniform density function depicted in the center plot. Conversely, the leftmost plot illustrates a poorly chosen $q(\boldsymbol{x})$ that would increase the variance in the estimator.

It is crucial to emphasize that the choice of q is an essential but non-trivial consideration. A poorly chosen q would induce a density ratio $\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}$ where the estimation of the expected value would have a high variance. Consequently, the sample average would have high uncertainty and would be determined by a small portion of the samples. The hope is that we have some understanding of $p(\boldsymbol{x})$ and $f(\boldsymbol{x})$ that could guide us in our selection. For instance, we may know where $f(\boldsymbol{x})$ is high or maybe even know a certain approximation of $p(\boldsymbol{x})$.

2.3.1 Importance Sampling in Deep Learning

The intuition for leveraging IS in DL could be motivated by the unknown distribution of the data we are dealing with. To illustrate this, let's consider an abstract example visualized in figure 2.12, which depicts the unknown data distribution for a binary classification task where we want to distinguish cats and dogs. Note that we have condensed the whole input feature space into a single variable $\boldsymbol{x}$ for visualization purposes. A typical trajectory when training a DL model on a task like this is the fast rate of convergence in the early stages of training. At these stages, the model becomes quite good at correctly predicting easy samples. Easy samples in this context could be samples where cat or dog features are prominent. On rare occasions, the model gets exposed to samples where e.g. a strange-looking cat has almost indistinguishable features from a dog or samples where the animal's features are rare. These are the more challenging samples and lie in the long tail regions of the distribution densities. As conveyed by the figure, in these regions, the densities are relatively small and the model will rarely get exposed to these types of samples if sampling uniformly at random. In fact, the model will have to run through far more samples to become good at predicting the challenging regions correctly as opposed to predicting the easy regions correctly. This motivates the utility of importance sampling. By oversampling the challenging regions, the model will be less subject to allocating compute resources on insignificant gradient contributions obtained from

uninformative samples and could instead focus on learning from more informative samples, which induce a greater albeit favorable change in the parameters. As such, the parameters of the model will possibly reach a local optimum with less computation. This is especially lucrative if the dataset is large and consists of a big proportion of uninformative samples that don't vary significantly in input feature space.

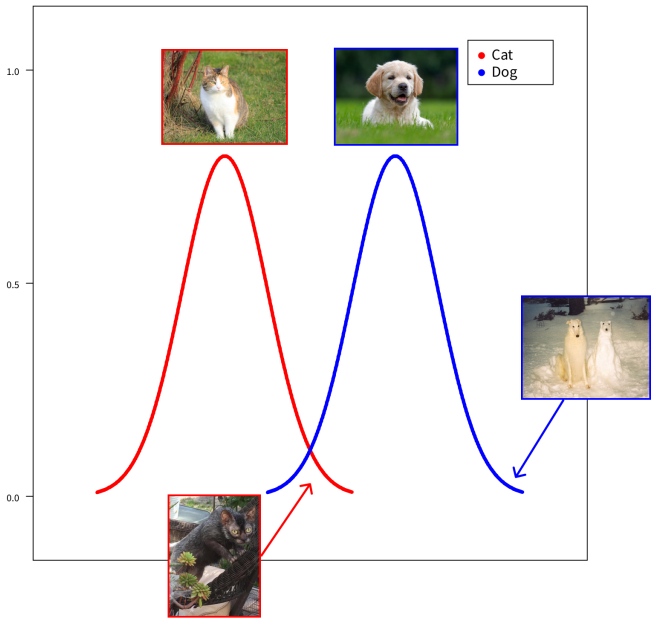

Figure 2.12: An illustration of a toy example: cat vs dog unknown data distributions. Note that the x-axis is an abstract and arbitrary representation of the input feature space. The lower middle animal image depicts a Lykoi, which is a cat breed that resembles a wolf or a dog.

To formally reason why importance sampling could be beneficial in DL, let L be a loss function, $\boldsymbol{\theta}$ the learnable parameters of a deep learning model, and $D = \{(\boldsymbol{x_i}, \boldsymbol{y_i})\}_{i=1}^{n}$ the data of which D_t is a batch from D at timestep t. A gradient descent update is expressed as the following:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \eta_t \boldsymbol{\nabla} \boldsymbol{L}_{\boldsymbol{\theta_t}}(D_t) \tag{2.7}$$

where η_t is the learning rate at timestep t. With this in mind, we can express the error term h_t at time t as the distance between the current parameters $\boldsymbol{\theta_t}$ and the optimal solution $\boldsymbol{\theta}^*$.

$$h_t = ||\boldsymbol{\theta_t} - \boldsymbol{\theta}^*||_2^2 \tag{2.8}$$

The convergence rate can now be defined as the difference in the errors between two consecutive iterations, i.e.

$$\begin{aligned} h_{t+1} - h_t &= \|\boldsymbol{\theta_{t+1}} - \boldsymbol{\theta^*}\|_2^2 - \|\boldsymbol{\theta_t} - \boldsymbol{\theta^*}\|_2^2 \\ &= (\boldsymbol{\theta_{t+1}} + \boldsymbol{\theta_t} - 2\boldsymbol{\theta^*})(\boldsymbol{\theta_{t+1}} - \boldsymbol{\theta_t}) \\ &= (2\boldsymbol{\theta_t} - 2\boldsymbol{\theta^*} - \eta_t \boldsymbol{\nabla L_{\theta_t}}(D_t))(-\eta_t \boldsymbol{\nabla L_{\theta_t}}(D_t)) \\ &= -2\eta_t(\boldsymbol{\theta_t} - \boldsymbol{\theta^*})\boldsymbol{\nabla L_{\theta_t}}(D_t) + \eta_t^2(\boldsymbol{\nabla L_{\theta_t}}(D_t))^2 \end{aligned} \tag{2.9}$$

The expectation over the batches gives us the expected contribution for a single iteration:

$$\begin{aligned} \mathbb{E}_p[h_{t+1} - h_t] &= -2\eta_t(\boldsymbol{\theta_t} - \boldsymbol{\theta^*})\mathbb{E}_p[\boldsymbol{\nabla L_{\theta_t}}(D_t)] + \eta_t^2 \mathbb{E}_p[(\boldsymbol{\nabla L_{\theta_t}}(D_t))^2] \\ &= -2\eta_t(\boldsymbol{\theta_t} - \boldsymbol{\theta^*})\mathbb{E}_p[\boldsymbol{\nabla L_{\theta_t}}(D_t)] + \eta_t^2((\mathbb{E}_p[\boldsymbol{\nabla L_{\theta_t}}(D_t)])^2 + \mathrm{Var}_p[\boldsymbol{\nabla L_{\theta_t}}(D_t)]) \end{aligned} \tag{2.10}$$

Now, reducing $\mathrm{Var}_p[\boldsymbol{\nabla L_{\theta_t}}(D_t)]$ in equation 2.10 implies improving the convergence rate, since the difference in the distances to the optimal solution should ideally be as negative as possible, i.e. we want $h_{t+1} << h_t$. This could be satisfied by biasing the sampling of D_t. In particular, when the aim is to enhance the convergence rate in the long tails of the data distribution where the data is sparse, it is in our interest to select a heuristic q that could outperform the uniform sampling in reducing the variance of $\mathrm{Var}_p[\boldsymbol{\nabla L_{\theta_t}}(D_t)]$ when D_t consist of such samples.

It is worth highlighting the challenge of estimating the importance of each sample as the model evolves. Specifically, when dealing with mini-batch gradient descent, the gradient estimates are obtained from mini-batches of data at certain parameter states. IS will enable the training phase to compile mini-batches with a high concentration of important samples for a given state of parameters. Because only a fraction of the training examples are sampled every iteration, the importance scores retrieved from previous observations may poorly reflect the current state of the model's parameters. It is of interest to consequently leverage an update rule $q_t(\boldsymbol{x}) \rightarrow q_{t+1}(\boldsymbol{x})$ which accounts for this change in importance, possibly even for non-sampled data points with e.g. smoothing [11].

In practice, this could more or less incur the following overarching procedure;

Algorithm 4 Simple importance sampling strategy in mini-batch gradient-descent training

Data: D
Model parameters: $\boldsymbol{\theta}$
$q \propto U(1, |D|)$
for t = 1,...T **do**
  $D_t \longleftarrow n_b$ data points sampled with q from D
  $[s, \boldsymbol{\nabla L}] \longleftarrow$ ForwardBackward$(D_t, \boldsymbol{\theta})$
  $\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} - \eta_t \cdot \boldsymbol{\nabla L}$
  $q \longleftarrow$ UpdateImportanceScores(q, s)
end for

Note that this pseudo-code is overly simplified and is far from representative of all IS strategies. In other words, several steps may substantially differ depending on the method, task, and heuristic. However, an interesting step is the stochastic process where entries in D_t are sampled with q, inferring that we can concentrate on the samples that tend to contribute more to the update of $\boldsymbol{\theta}$. Consequently, accelerating training and spending fewer computations on less informative samples.

3

Related Work

Estimating the importance of each sample to enable accelerated training is a concept also present in other techniques such as curriculum learning [6] and core-set selection [2]. Similarly to IS, curriculum learning prioritizes certain samples during training to ultimately enhance the training procedure. However, the difference being that curriculum learning aims at prioritizing easy samples in the beginning and progressively introduces more challenging samples as the training advances. While the benefits of curriculum learning has been demonstrated by a significant amount of empirical data [37], the method requires defining the order of the samples before training. This is in itself a costly task, as the sample difficulties must have been assessed beforehand. As tedious as it is, it is commonly done either manually or by transferring knowledge from a pre-trained model. However, self-paced learning [38] alleviates this problem and reformulates this ad-hoc scheme by utilizing training dynamics to decide if a sample is yet too challenging and sorts the data according to that heuristic.

On the other end of the spectrum are methods related to core-set selection. These methods aim at finding a subset of the most informative samples in a labeled or unlabeled dataset. Applications include querying the most important samples to label or pruning data for budgeted and/or accelerated training. Toneva *et al.* [1] propose pruning data based on the number of forgetting events. The authors show that samples that are easily forgotten by the model during training tend to be more informative, while samples that are rarely forgotten tend to be less important. Excluding rarely forgotten examples and training the model on a fraction of the dataset may therefore improve convergence without degrading the model. Pleiss *et al.* [39] achieve better final performance by excluding mislabeled or overly ambiguous samples. This was achieved using a classification-based metric called margin [40] [41] [42], in their case, the final layer margin. The final layer margin is computed as the difference between the ground truth logit and the highest logit among the other classes, i.e. $M(\boldsymbol{x}, c) = \underbrace{z_c(\boldsymbol{x})}_{\text{assigned logit}} - \underbrace{\max_{i \neq c} z_i(\boldsymbol{x})}_{\text{largest other logit}}$, where c is the index of the ground truth value in the vector, and $z(\boldsymbol{x}) \in \mathrm{R}^c$ is the pre-softmax output (logits), from the model given input x.

Paul *et al.* [4] suggest using the error vector as an importance heuristic to prune data after merely a few epochs of training. Sener and Savarese [2] propose a method which involves optimizing a k-center problem. The key idea is to find a subset of so

called center points in the data that when trained on, achieve a performance as close as possible to the one that would be obtained when training on the whole dataset. With a distance heuristic, the algorithm tries to choose b center points such that the largest distance between a data point and its nearest center point is minimized. For computing the distance between samples, the authors suggest using the activations of the final fully-connected layer. While the notion of informative samples is shared between IS and core-set selection approaches, core-set selection typically requires a full training phase or at least a pretraining stage before being able to assign sample scores.

IS can be seen as a middle ground approach of the two aforementioned concepts [43]. Like curriculum learning, IS involves prioritizing samples online during training. Like core-set selection, IS needs to assign samples' importance scores dynamically. Various suggestions for measuring importance and how to use importance sampling in a training algorithm have been proposed in the literature. Some popular suggestions have been to sample training examples based on the loss. Loshchilov and Hutter [12] employ a strategy where datapoints are ranked with respect to their latest loss and then sample based on an exponentially decaying function applied on top of the ordered samples. Ioannou *et al.* [22] suggest an approach where at the end of each epoch, a portion of the lowest loss yielding samples are replaced with a portion of the highest loss yielding samples. At the cost of introducing a hyperparameter, the authors also suggest utilizing a momentum component for the importance score for a more balanced sampling scheme and a prominent improvement in convergence speed. The paper states that this approach achieves up to 40% convergence speed-up on CIFAR10 [44]. Katharopoulos and Fleuret [45] propose a strategy that trains a second neural network in parallel and predicts the loss of the main model for the sampling procedure. They call this method approximative importance sampling. This study [45] claims a speed-up of 20% to 30% on image classification on MNIST [46] and CIFAR10. However, it is important to note that the authors evaluate the speed-up based on the convergence in training loss which should be biased in this setting. Katharopoulos and Fleuret [7] later published a study where they leverage an upper bound on the gradient norm instead. In fact, it has been theoretically shown that the optimal proposal distribution should be proportional to the per-sample gradient norm [10]. However, the per sample gradient norm is computationally prohibitive to compute, so it may be impractical to apply in practice. Nonetheless, the authors suggest that the gradient of the loss with respect to the pre-activation outputs of the last neural network layer gives an upper bound to the gradient norm of all the parameters, and can be utilized in IS to converge faster to a lower test error.

In contrast to the aforementioned studies, other studies have shown compelling evidence for importance samplers that concentrate on uncertain samples rather than purely high loss samples. Chang *et al.* [11] compared Loshchilov and Hutter [12]'s approach with samplers that, instead of using loss, focus on samples with high prediction variance or threshold-closeness. Their observations concluded that preferring high loss samples works well when the task is relatively easy (i.e. both training and testing accuracy are high). However, when the task is complex and the dataset

is challenging and/or noisy, a preference towards uncertain examples showed to be more beneficial for the final generalization performance.

Online batch selection A popular method for addressing the problem of a rapidly evolving importance distribution involves doing additional forward passes. Some of the aforementioned studies employ this concept and several other works have explored this route [7], [9], [12], [47], [48]. They motivate its advantage over the naive sampling from Algorithm 4 when measuring convergence based on wall-clock time instead of the number of forward propagations. The idea is to sample a big batch to forward propagate. From this, a sample heuristic could be utilized to select a subset for the backward-pass. Assuming that the backward pass is the bottleneck in the algorithm, this approach could potentially result in a speedup. However, for this, many samples have to be loaded from disk into memory and decoded by the Central Processing Unit (CPU) at a time, which in itself may be a greater bottleneck than the backward-pass when working with large images. For this reason, online batch selection does not appear useful in OD for AD and is not investigated in this thesis.

Reducible holdout framework Mindermann *et al.* [9] proposed Reducible Holdout (RHO) which is a framework that is used in combination with IS. It first trains a smaller model on the same task using a held-out part of the dataset denoted as D_{ho} that is consequently excluded from the main dataset. This smaller model is trained once, after which it performs one forward-pass over the whole dataset. Using the forward-pass and a heuristic such as the loss, a score for each sample is computed. They explore the loss as an heurstic. The loss produced from the small model is denoted the irreducible loss. When training the real model, this irreducible loss is subtracted from the normal loss produced by the main model to form a new heuristic called the reducible loss. The formal notation for this importance heuristic is simply:

$$\overbrace{\underbrace{L(\boldsymbol{x}, \boldsymbol{y}; D_t)}_{\text{training loss}} - \underbrace{L(\boldsymbol{x}, \boldsymbol{y}; D_{ho})}_{\text{irreducible holdout loss}}}^{\text{RHO loss}} \tag{3.1}$$

The general intuition behind RHO loss selection relies on the empirical observation that it avoids easy, noisy, and out-of-distribution samples. Noisy samples are challenging but uninformative. Such samples include samples with incorrect labeling and may contribute to degrading the model's performance. Out-of-distribution samples are in other words outliers. Despite not being mislabeled, from the perspective of the test data evaluation, it could be hurtful for the model to put too much emphasis on these samples.

The following points are stated in the paper:

- Since easy samples tend to have a low training loss, and the RHO loss is always less than the training loss, such samples are less favored. If the model were

to forget an easy sample, the metric would account for this since the training loss would be high, and which increases the chance of revisiting it.

- Noisy samples would be harder for the holdout model to generalize, incurring a high irreducible loss and consequently a low RHO loss.
- Out-of-distribution samples follow a similar reasoning as for the noisy samples. Consider an outlier $\boldsymbol{d\prime}$ and a non-outlier $\boldsymbol{d}$. Since D_{ho} is an unbiased split from D, the following holds true for both datasets $P(\boldsymbol{d\prime}) < P(\boldsymbol{d})$. Naturally, the holdout model will perform worse on out-of-distribution samples and induce a high irreducible loss and low RHO loss.

Several of the aforementioned studies suggest that accelerated training can be obtained by exploiting informative samples using importance sampling. However, none of them show any empirical evaluations in the task of OD, which is a seemingly more complex task when compared to image classification. Not only is the complexity of OD higher than the tasks that IS has been reported with, but the dataset [30] used in this thesis is vastly more complex than CIFAR10 and CIFAR100 which is among the popular choices in these studies. Previous studies have investigated other ideas for enhancing training convergence and also try to address the long tail problem in the context of DL OD [49]. These methods are generally concerned with some function for selecting the most important regions within images rather than operating on a dataset level and often require modifications to the optimization problem by introducing some component to the loss or the backpropagation stage. Online Hard Example Mining (OHEM) by Shrivastava *et al.* [49] has shown to be one of the more prominent methods where the idea is to consider only the most beneficial regions of interest for the backpropagation. This method is, however, limited to two-stage object detectors as the regions are parsed from a RoI pooling stage. Based on the idea of OHEM is a method called Loss Rank Mining by Yu *et al.* [50]. The method can be applied to one-stage detectors and works similarly by filtering out all but the k highest prediction-wise losses from the final feature map. Nevertheless, this approach is mainly tailored to grid-based and template-based detectors and does not translate to a stable solution for detectors where the image plane stride of the dense prediction map is low. The culprit is the fixed k along with the fact that the number of positive target pixels can vary substantially between samples. Another method worth mentioning is Prime Sample Attention (PISA) [51]. It presents an intermediate preference between hard mining and random sampling of image regions. While it does propose an importance-based sampling scheme, it does not, similar to the other aforementioned methods that are applicable in OD, operate on a dataset level.

4

Methods

This thesis studies the possibility of leveraging importance sampling to accelerate training in DL OD. Given this objective, the project branches into two main areas of practice: 1) The development of a sufficiently performing, low-latency object detection system along with its full data preprocessing, training, evaluation, and inference pipeline, and 2) The development of an IS framework for applying IS methods and conducting relevant experiments. The methods presented in this chapter heavily rely on theory provided in chapter 2 and related work from chapter 3. However, due to the absence of literature on IS in OD, related work has been adapted from a predominantly image classification setting to an OD setting. This discrepancy alone has revealed multiple challenges, which, together with other applied topics, are presented in this chapter in more detail.

The methods outlined in the sections below first cover details about the object detection system used in the project. Subsequently the IS framework and design choices for the system as a whole are presented.

4.1 Object Detection

This thesis employs a single-stage anchor-free object detector that closely resembles FCOS [35], presented in section 2.2.5. Similarly to FCOS, the model used in this project is fully convolutional and directly computes its predictions on dense location-aware feature maps where all the predictions for classification and regression for a location (x, y) are stacked along the channel axis as illustrated in figure 4.1.

4.1.1 Backbone

Tian *et al.* [35] use ResNet50 and ResNet101 as backbones, i.e. ResNet models with 50 and 101 convolutional layers respectively. This thesis uses a ResNet34 where each convolutional layer has only 50% of the channel count of the original model proposed by He *et al.* [52]. This is due to the constraints of AD, where the model has to be small in order to run in real-time in a car while not compromising too much performance and to accommodate the size of the dataset to minimize the risk of severe overfitting.

4.1.2 Architecture

Tian *et al.* [35] use an FPN in FCOS, in order to handle differently scaled objects. This project instead uses Deep Layer Aggregation (DLA) [53] where the last 3 stages are aggregated to go from 32x to 8x down-sampling. Hence, a single feature map is returned from the model. This keeps the architecture simple and the scale invariance in latent feature space. For this thesis, it is deemed secondary to achieve state-of-the-art performance, and simplicity is prioritized to have a clean environment for studying IS.

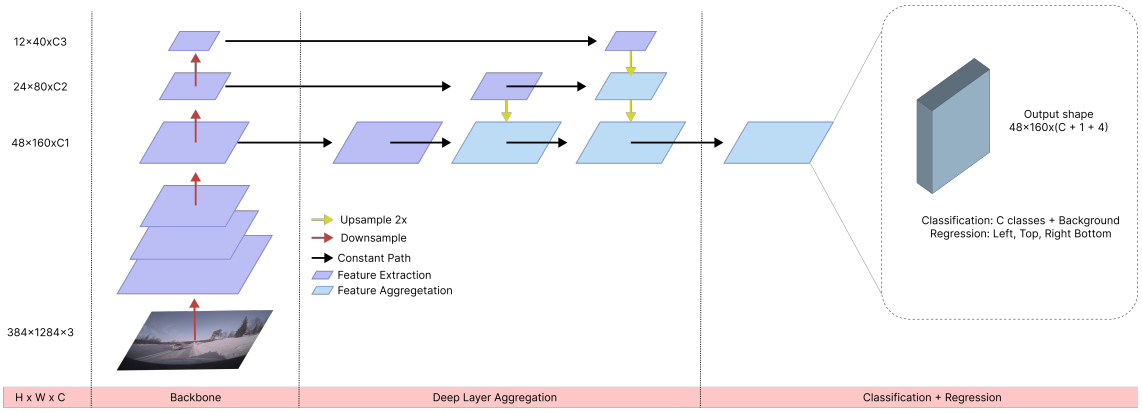

Figure 4.1: An abstract overview of our model's architecture. The ResNet backbone extracts features from the input image, while the Deep Layer Aggregation module aggregates the extracted feature maps to produce the output in the form of a dense prediction map.

4.1.3 Don't-Care Tokens

In a dataset, there may be objects whose class is unknown, so called "don't-care" objects. This label can for example represent an object where the annotator could tell that there was an object, but could not tell which class it belongs to. For this kind of object, it is desired to only propagate gradients for the bounding box, but not for the classification part. To accomplish this, a mask is defined during data processing, which marks any pixel that is contained by a don't-care object. The mask for these pixels is 0, and 1 otherwise, and the mask is pixel-wise multiplied by the classification loss in order to only propagate gradients of pixels where the class is known.

4.1.4 Loss Function

In their paper, Tian *et al.* [35] use equation 2.5 as the loss function. Instead of using focal loss, or training C binary classifiers, that is one for each class, this thesis uses a Softmax activation function followed by categorical cross-entropy loss. Softmax is in this context expressed as $\sigma(z_{x,y,i}) = \frac{e^{z_{x,y,i}}}{\sum_{j=1}^{C} e^{z_{x,y,j}}}$, where $z_{x,y,i}$ is the preactivation logit for the classification prediction at location (x, y) and class index i. As such, the output will represent a probability vector which conforms to the fact that an object

can only belong to one of the classes. Categorical cross-entropy loss is expressed as follows;

$$L_{cls}(\boldsymbol{c}_{x,y}, \boldsymbol{c^*}_{x,y})) = -\sum_{i=1}^{C} \mathbb{1}_{i=c^*_{x,y}} \cdot \log(c_{x,y,i})$$

.

Additionally, it is worth noting that the original FCOS loss function does not consider don't-care tokens. Consequently, the loss function in this project is as follows;

$$\begin{aligned} L((\boldsymbol{c}_{x,y}, \boldsymbol{u}_{x,y}), (\boldsymbol{c^*}_{x,y}, \boldsymbol{u^*}_{x,y})) = &\frac{\lambda_{cls}}{N_{care}} \sum_{x,y} \mathbb{1}_{care_{x,y}} L_{cls}(\boldsymbol{c}_{x,y}, \boldsymbol{c^*}_{x,y}) \\ &+ \frac{\lambda_{reg}}{N_{pos}} \sum_{x,y} \mathbb{1}_{c^*_{x,y}>0} L_{reg}(\boldsymbol{u}_{x,y}, \boldsymbol{u^*}_{x,y}) \end{aligned} \tag{4.1}$$

where

- λ_{cls} and λ_{reg} are weights for the classification and bounding box regression losses respectively (and are set to 1 unless otherwise specified).
- $\mathbb{1}_{care_{x,y}}$ is the don't-care mask value at pixel (x, y), as described in section 4.1.3.
- N_{care} is the number of positive pixels in the don't-care mask, i.e. $\sum_{x,y} \mathbb{1}_{care_{x,y}}$.

4.1.5 Non-Maximum Suppression

Since every pixel in an image predicts a bounding box and crowds the region proposals, a method is required to filter the most relevant predictions. There are two kinds of bounding boxes that need to be filtered away: low-confidence bounding boxes, and bounding boxes predicting the same object as another bounding box. As described in Section 2.2.4, NMS takes care of both of these cases and is used during inference in this project. The confidence- and IoU-thresholds are both set to 0.5.

4.2 Importance Sampling

For the evaluation of several IS methods to be convenient, a framework is developed which allows for customization of the strategy. This framework works by keeping a list of importance scores, one for each training example. Whenever the model is going to train, one or more examples are sampled using the importance distribution. The importance scores are all positive, and the sampling is done by sampling an example with a probability proportional to its importance score:

$$p_i = \frac{s_i}{\sum_{j=1}^{N} s_j} \tag{4.2}$$

where s_i is the importance score of sample i and N is the number of training samples.

After the model has trained on these examples, the importance scores of the selected examples are updated using the outcome of training on them. The outcome of

the training refers to any attribute which can be computed from a single forward and backward pass, e.g. loss, prediction error, gradient norm etc. How the new importance score is computed given these attributes differs, and is here called a sampling heuristic. Various heuristics are explained in this section. There are also strategies that are invariant to which heuristic is used, but which influence the sampling in other ways.

4.2.1 Baseline

The baseline approach is to not use any importance sampling at all. In this approach, the order of the training examples is randomized before every epoch to remove any bias in how the data is ordered. The model then trains on every training sample exactly once per epoch. This approach serves as a baseline to compare all other approaches to and is referred to as either "baseline" or "sequential" in the rest of the report.

4.2.2 Smoothing

Smoothing is a heuristic-invariant approach that can serve two purposes: 1) Induce more exploration, and 2) Reduce the effect of stale importance values. It is important to note that smoothing can either be accumulated after each iteration or simply used as a non-accumulating bias term to the importance score. When the probabilities are calculated as in equation 4.2, the distribution is closer to uniform than it would have been otherwise, which induces more exploration since all samples' importance scores are increased, not just the ones that were selected and trained on. In this thesis, smoothing acts as a non-accumulating bias term and the value is dynamically set to the mean importance divided by the number of examples in the dataset:

$$\text{smoothing} = \frac{1}{N^2} \sum_{i=1}^{N} s_i \tag{4.3}$$

4.2.3 Loss-based Importance Sampling

The Loss Importance Sampler (LIS) is the first IS heuristic. The importance of a sample is simply equal to the loss of that sample; $s_i = L_{\boldsymbol{\theta}}(\boldsymbol{x_i}, \boldsymbol{y_i})$. As conveyed in chapter 3, the loss is a common IS heuristic and has been used by many practitioners. However, when dealing with the loss of an object detector, it is important to be mindful of the differences between the optimization problem and the indirect goal of the importance sampler. Recall that the loss in this project produces a pixel-wise loss in the dense prediction tensor. Since the aim is to accelerate the training of detecting objects, computing an importance score that is homogeneously reduced per pixel is counterproductive, because the importance scores can be dominated by large objects (i.e. objects that occupy a great number of pixels) along with negative locations (i.e. background).

4.2.4 Object-wise Reduction

One way of addressing the above-mentioned problem is to reduce the scores per object. This is done by keeping track of which pixels belong to each object and averaging the values in an object-wise manner. This yields one importance value per object, which can then be reduced further into a sample-wise value. Some suggestions for the second reduction are as follows:

- Sum: The object-wise importances are summed.
- Average: The object-wise importances are averaged.
- Area-weighted mean: The object-wise importances are scaled by some value proportional to the area of their objects, and the averaged. They can for example be scaled by the square root of the areas.
- Max: Pick only the highest of the object-wise importances to represent the sample.
- Root mean square: Compute the root of the mean of the squares of the object-wise importances.

This thesis considers mainly the average as an object-wise reduction technique, which is the one that is referred to unless otherwise stated. An illustration of how the average reduction works can be seen in figure 4.2.

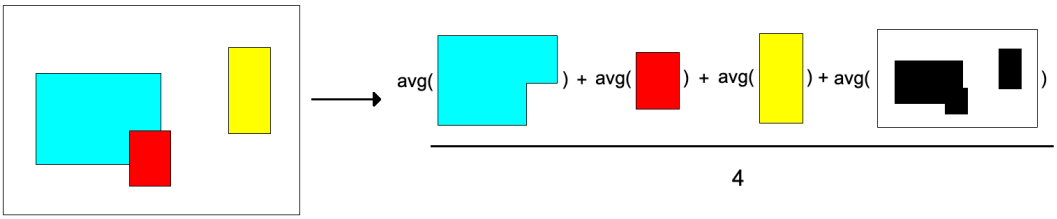

Figure 4.2: Visual representation of the average object reduction. Note that the set of background pixels is also considered to be an additional object.

The average is prioritized in this project because it is the most stable of the reductions, and the other methods have effects that might be undesired. For example, using the sum would generally give high scores to samples with many objects, and using the area-weighted average would generally give high scores to samples with large objects. The average is however not perfect, as it risks drowning out important objects if there are many unimportant objects in the same sample. The max reduction would not do this, but is not as stable and would be susceptible to noise.

4.2.5 Error-based Importance Sampling

The error-based sampler assigns a score scaled with how poorly the model performs in its prediction about each sample. This is similar to how the loss sampler works. One of the original methods is presented by Chang *et al.* [11]. They compute a

score that depends on the history of an error-based heuristic over the epochs. For classification the error-based heuristic is $\frac{1}{M}\sum_{t=1}^{M}(1-\boldsymbol{y}^T\hat{\boldsymbol{y}}_{\boldsymbol{t}})$, where M is the current epoch. Paul *et al.* [4] adopt a similar heuristic but compute the score as the norm of the error vector and do not average the score over multiple epochs. The error-based sampler used for this thesis is similar to the approach from Chang *et al.* [11], but does not consider the history, and is adapted in several ways to include both the multi-modality of OD and the dense output feature map of the FCOS approach.

For the classification part, the score for a sample is defined as

$$\frac{1}{N_{care}}\sum_{x,y} -log(\boldsymbol{c^*}^T_{x,y}\boldsymbol{c}_{x,y})\cdot \mathbb{1}_{c^*_{x,y}>0} \tag{4.4}$$

The score for the bounding box regression task is similar but uses the per-pixel IoU-loss, and the regression mask instead of the classification (don't-care) mask. These two components, classification and regression, each being a masked average, are then summed, yielding a single value representing the poorness of the prediction for the sample. In the case that object-wise reduction is desired, it can be used instead of a masked average. Note how the calculations, similarly to cross-entropy, use the negative logarithm to enable an importance score that could range from zero to infinity. In the case of an averaged object-wise reduction, this trick could alleviate the problem of hard underrepresented classes getting drowned out by other easy classes within the same image.

4.2.5.1 Uncertainty-based Importance Sampler - Decision Boundary Closeness

It can be beneficial for the model to train not just on samples that it makes big mistakes on, but samples where the model is uncertain, i.e. the confidence of a prediction is close to the decision boundary. In this case, the computations of the two components of the error can be modified to instead represent the model's uncertainty. Let $a_{x,y} = \boldsymbol{c^*}^T_{x,y}\boldsymbol{c}_{x,y}$. Then, replace $a_{x,y}$ with $4\cdot a_{x,y}\cdot(1-a_{x,y})$. The classification part of the score for a sample is now

$$\frac{1}{N_{care}}\sum_{x,y} -log(4\cdot a_{x,y}\cdot(1-a_{x,y}))\cdot \mathbb{1}_{c^*_{x,y}>0} \tag{4.5}$$

Since $a_{x,y}\in[0,1]$, $a_{x,y}\cdot(1-a_{x,y})$ will reach its highest point when $a_{x,y}=0.5$; at its decision boundary. The expression is multiplied by 4 simply to normalize it back into the range $[0,1]$. This technique is applied similarly for the regression task but with IoU-loss instead, which is also in the range $[0,1]$. We denote this sampler Uncertainty Importance Sampler (UIS), and it is analogous to the threshold-closeness-based sampler from Chang *et al.* [11] which was shown to outperform loss-based approaches. Similarly to the loss-based sampler UIS uses the object-wise reduction instead of the masked average in equation 4.5.

4.2.6 Reducible Hold-out Loss

The loss value of a noisy or out-of-distribution sample can be very high, so using it directly would give high importance to such training examples, which is undesirable.

For this reason, the RHO-framework is considered. A discrepancy between the original paper from Mindermann *et al.* [9] is that they use online batch selection, which this thesis does not. Another difference is how the reducible losses are computed. The authors compute the reducible loss as the difference between the training loss and the irreducible loss, see equation 3.1. In this thesis, the reducible losses are produced in a similar way but are also shifted and re-scaled:
Let $\boldsymbol{s} = L(\boldsymbol{x}, \boldsymbol{y}, D_t) - L(\boldsymbol{x}, \boldsymbol{y}, D_{ho})$. Then

$$\hat{s}_i = \frac{s_i - min\ \boldsymbol{s}}{max\ \boldsymbol{s} - min\ \boldsymbol{s}} \tag{4.6}$$

This is done in order to keep the reducible scores positive and compatible with equation 4.2. Finally, the authors sample training examples by picking the highest reducible score. This is problematic when not using online batch selection, as some of the importance scores can be stale. In this project, the training samples are instead sampled with the probabilities given in equation 4.2. Hyperparameters revolving around the holdout-training are given in table 4.1 below.

Table 4.1: Hyperparameters for the holdout-training in our experiments.

Hyperparameter	**Value**
Backbone	Resnet 34
Feature multiplier	0.25
Num epochs	100
Batch size	128
Holdout training split	0.15

4.2.7 Rejection-based sampling

Estimating each sample's importance while the model's parameters evolve is a challenge that has been pointed out in numerous previous studies [7], [9], [43], [48]. Outdated importance scores may decrease the effectiveness of importance sampling as the proposal distribution in such a case poorly reflects the target importance distribution. This problem is often addressed using online batch selection, which appears inapplicable for this project for reasons discussed in chapter 3. Apart from smoothing, to further alleviate this issue, we explore a method analogous to a random walk in the dataset. The idea is to accept or reject samples based on a comparison between one of the latest forward-propagated samples and a randomly drawn proposed sample. This is done instead of sampling naively based on the scheme highlighted in algorithm 4. Although the proposed sample may have an outdated importance score, at least the latest forward propagated sample should be up-to-date. For this approach, there is a lack of theoretical guarantees. It is however an intriguing approach to investigate empirically as a means to explore alternative sampling strategies. The algorithm in question is outlined as follows:

Algorithm 5 Rejection-based sampling scheme as an alternative to the naive sampling scheme in algorithm 4

Data: D
Batch size: n_b
Model parameters: $\boldsymbol{\theta}$
$q_0 \propto U(1, |D|)$
$D_0 \longleftarrow n_b$ data points sampled with q from D
for $t = 1, ..., T$ **do**
  for $i = 1, ..., n_b$ **do**
    Sample candidate sample $\boldsymbol{x_c} \sim q$
    Get active sample $\boldsymbol{x_a} \longleftarrow D_{t-1,i}$
    if $q(\boldsymbol{x_a}) > 2 \cdot q(\boldsymbol{x_c})$ or $q(\boldsymbol{x_c}) < p \cdot (q(\boldsymbol{x_c}) + q(\boldsymbol{x_a}))$ where $p \sim U(0, 1)$ **then**
      Reject candidate sample $D_{t,i} \longleftarrow \boldsymbol{x_a}$
    else
      Accept candidate sample $D_{t,i} \longleftarrow \boldsymbol{x_c}$
    end if
  end for
  $[s, \boldsymbol{\nabla L}] \longleftarrow$ ForwardBackward$(D_t, \boldsymbol{\theta})$
  $\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} - \eta_t \cdot \boldsymbol{\nabla L}$
  $q \longleftarrow$ UpdateImportanceScores$(q,\ s)$
end for

4.3 Regression-agnostic sampling

As the results in chapter 5 will show, some sampling strategies intrinsically prioritize and improve performance for certain sizes of objects, even when using object-wise reduction. For this reason, it is interesting to completely ignore regression in the importance calculations. In the heuristics presented above, the importance scores are always computed as the sum between a classification part and a regression part. When ignoring regression, the regression part of the equation is simply set to 0.

4.4 Dataset

The ZOD [54] is a recently released open dataset from Zenseact [30]. ZOD includes many modalities useful for AD, e.g. road semantic segmentation, and 2D- and 3D bounding box detection in both single frames and sequences. For this thesis, only 2D bounding boxes for dynamic objects in single frames are used. This includes approximately 90,000 training images and 10,000 validation images.

As can be seen in figure 4.3, the license plates for cars in ZOD are anonymized, and this is also the case for faces. ZOD gives two types of anonymization; blurring and deepfakes. However, the anonymization is automatic and imperfect, and occasionally leaves images visually corrupted. In order to use as clean a dataset as possible, this thesis uses the original, non-anonymized version of the data, which is not available publicly.

Since ZOD consists of frames with a resolution of 3848 × 2168, to streamline the experiments and minimize data-loading bottlenecks, the images are cropped in the data preprocessing pipeline. The Region of interest (RoI) is approximately the middle third in the x-dimension and the middle sixth in the y-dimension. This crop is responsible for objects far away in a typical scenario, so it is called the far Region of interest (RoI), and this version of the dataset is called ZODFAR. Figure 4.3 depicts how the crop is applied.

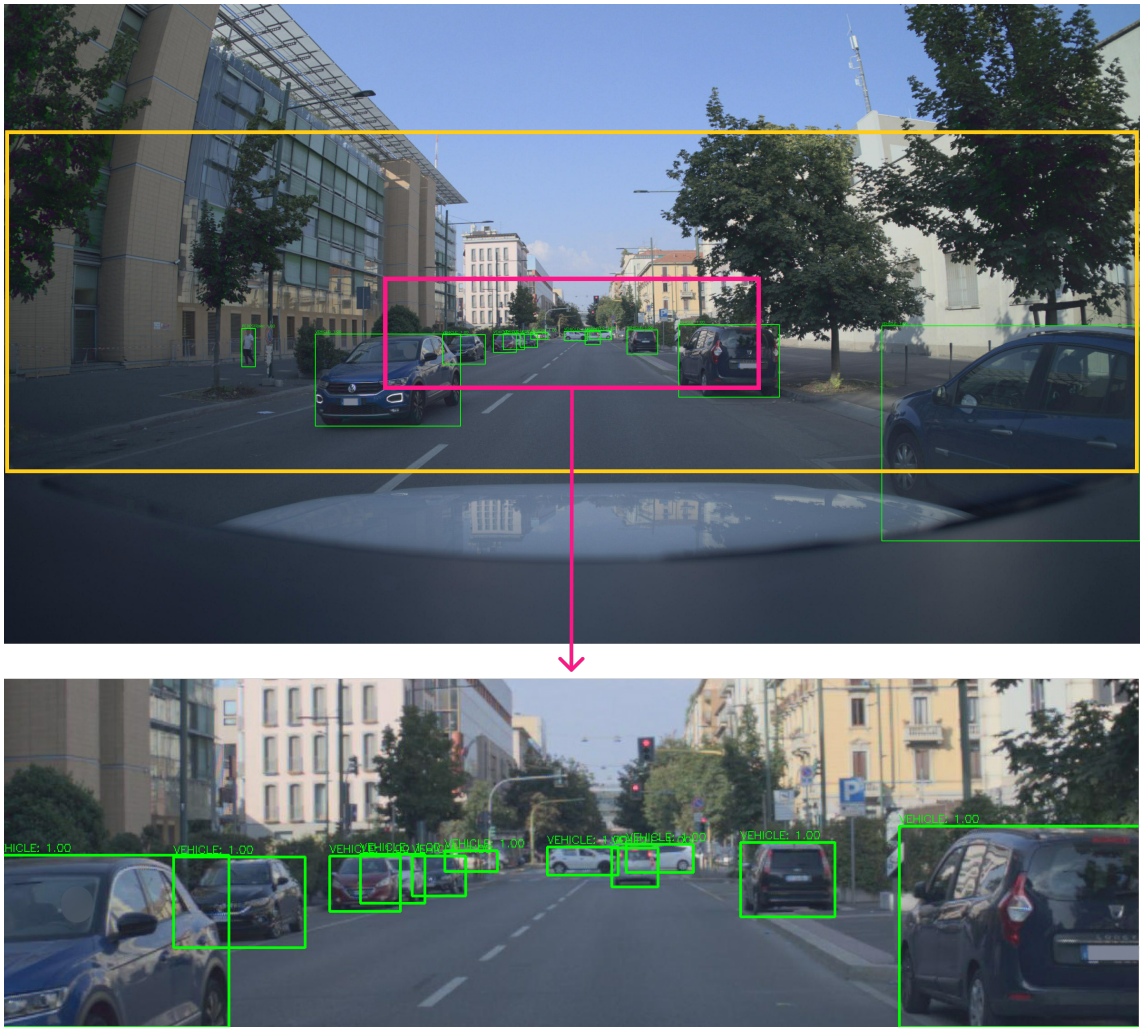

Figure 4.3: Visualization of a sample with its annotations. The orange box highlights a wide RoI while the pink box highlights a far RoI. Although both RoIs are important in a real scenario, only the far RoI is used in ZODFAR as a means to keep the system simple. The original image has image dimensions: 3848 × 2168 while the processed image has dimensions: 1280 × 384.

There are several reasons why this is the region of interest. Firstly, it reduces the size of the input image to the network by almost a factor of 20, which is necessary to keep the model running in a streamlined setting. Secondly, cropping a small part of the image is preferred to cropping a large part of the image and scaling it down, which leaves a lot of the bounding boxes extremely small in comparison to the total

image size.

The number of objects in ZODFAR are stratified by class and size in table 4.2. Small objects are defined as objects with an area of $< 0.33\%$ of the image size, medium objects have an area between 0.33% and 3%, and large objects have an area $> 3\%$ of the image size. These thresholds are mainly based on COCO's size stratifications [55].

Table 4.2: Number of objects in ZODFAR's dataset stratified by size and class. The number to the left of the parenthesis is the number of objects in the training set, while the number inside the parenthesis is the number of objects in the validation set.

	Pedestrian	Vehicle	VulnerableVehicle	Animal	DontCare
large	2342 (326)	93084 (11435)	2120 (353)	61 (5)	0 (0)
medium	32295 (3936)	217714 (21478)	11315 (1614)	578 (79)	0 (0)
small	66581 (7563)	193465 (24913)	12942 (1736)	1050 (135)	85999 (9168)

4.5 Experiment Details

Each sampling strategy was run several times (10 unless otherwise specified) and all runs of the same kind were averaged at each time step in order to give a robust estimate of the performance. Each run used a batch size of 128 and was trained for 350 epochs with 89976 training samples on an Nvidia A100 GPU 40GB, and took roughly 40-50 hours per run until complete. Note that since our system is running compute-intensive operations to log and collect certain data for monitoring and summaries of each iteration and validation, 40-50 hours might not be representative of the time it would take in a streamlined setting. Also note that for the runs with importance samplers, the notion of an epoch disappears after the first epoch is complete since they no longer iterate over the data sequentially. Lastly, all object detectors were trained using the Adam optimizer [18], with an initial learning rate of $\eta = 10^{-3}$, $\beta_1 = 0.9$ which is the exponential decay rate for the 1st moment estimates, $\beta_2 = 0.999$ which is the exponential decay rate for the 2nd moment estimates, and a numerical stability constant $\epsilon = 10^{-7}$.

5

Results

In this section, we present the results of our project. As stated, the purpose of this thesis is not to achieve state-of-the-art performance in OD, but instead to evaluate IS methods in terms of how much they speed up training in comparison to the baseline. As such, the results focus on the performance influenced by importance samplers and their various configurations. The results are presented with different levels of stratification, ranging from general mAP to AP stratified by both object size and class. The AP is computed over all recall values, and with interpolated precision values. mAP_{small}, mAP_{medium}, and mAP_{large} are also computed, meaning the same computation as above but filtered on small, medium, and large objects respectively.

The performance of an IS method is primarily measured by how much faster in terms of the number of iterations it could reach the maximum average mAP of the baseline, as in figure 1 in [9]. Although wall-clock time is what one ultimately wants to decrease, measuring speedup based on wall-clock is not meaningful in practice as it can depend on miscellaneous implementation details. In particular, speed-up is based on the rate of learning and can be calculated as follows;

$$\frac{\text{num\_iterations}_{\text{baseline}}}{\text{num\_iterations}_{\text{other\_method}}} - 1 \tag{5.1}$$

where

- $\text{num\_iterations}_{\text{baseline}}$ is the iteration where the baseline reaches its maximum mAP
- $\text{num\_iterations}_{\text{other\_method}}$ is the iteration where the mAP of the other method surpasses the maximum mAP of the baseline

Furthermore, as mentioned and as a means to delve deeper into the detection performance of certain subgroups in the data, we assess how the importance sampler affects performance across sizes and classes. It is especially interesting to observe how IS affects the performance in data points that lie in the low-density regions of the unknown data distribution. Since these types of samples are so scarce in the data, generalizing to such samples could require significantly more training iterations when sampling the batches uniformly without replacement. It is in our best interest to accelerate training on these types of samples, as they could bottleneck

the finish point of training. As a proxy for these types of samples, we will evaluate the performance of the *Animal* class in greater detail. The *Animal* class is vastly underrepresented in the data, as shown in table 4.2.

5.1 Loss and Uncertainty Samplers

In figure 5.1 we can see that for an average over 10 runs, sampling training examples using the mean-object-reduced UIS is beneficial and speeds up training by 95%, meaning that the training time could be almost halved if further performance is not required. On the other hand, sampling based on mean-object-reduced LIS does not appear to accelerate training. It is worth noting that measuring the speedup accordingly may be subject to volatility and is best assessed with a significant amount of runs averaged together. However in this case, the plot in figure 5.1 indicates a consistent and clear superiority of the UIS when compared to the baseline.

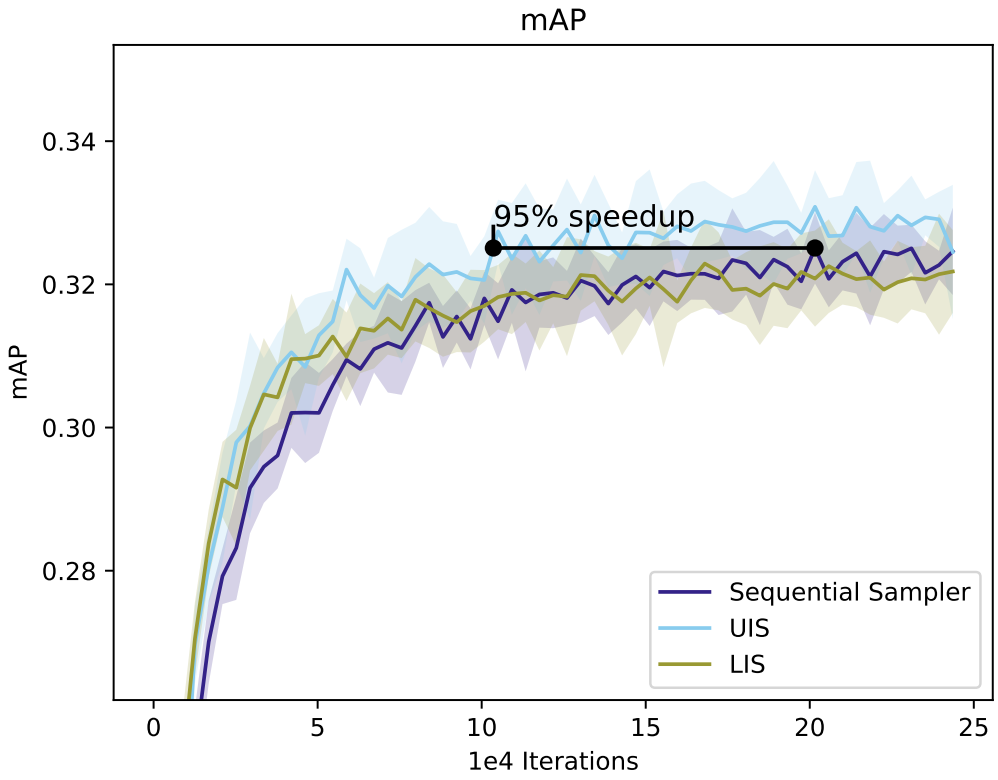

Figure 5.1: Comparison of overall *mAP* on ZODFAR for models trained with Sequential (baseline), LIS, and UIS. The closed black line marks the speedup for achieving the baseline's highest averaged mAP. The filled-in area marks ±1 standard deviation.

Looking closer at the results, specifically comparing performance on the different sizes of objects in figure 5.2, it is clear that both sampling approaches improve the

learning for learning small objects. This is also where the model performs worst, as indicated by the mAP for small objects being lower than for medium or large ones. For medium objects, the UIS appears to outperform the baseline, which the LIS does not. For large objects, neither sampling strategy reaches the highest mAP of the baseline, so no speedup can be measured.

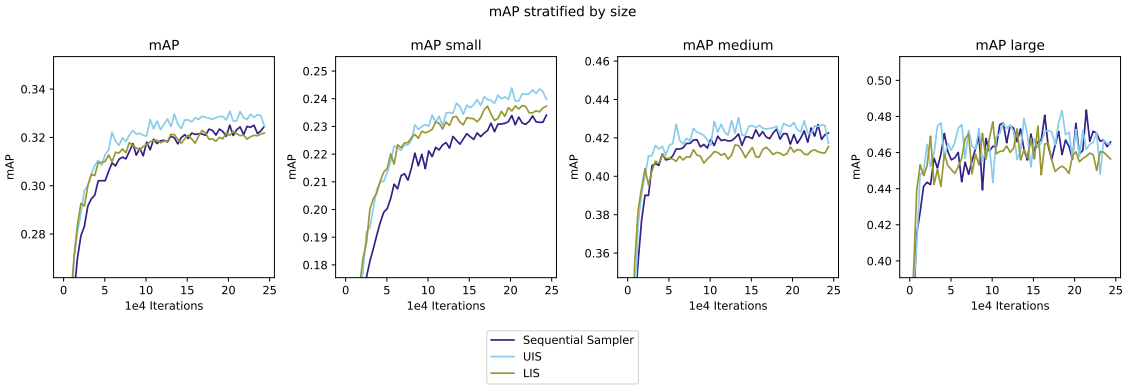

Figure 5.2: Comparison of mAP, mAP_{small}, mAP_{medium}, and mAP_{large} on ZODFAR for models trained with Sequential (baseline), LIS, and UIS.

Figure 5.3 reveals the performance of the different samplers stratified by both class and size. The UIS outperforms both the baseline and the LIS in the majority of categories and only degrades the model noticeably in $AP_{large,vulnerablevehicle}$. It is outperformed by the LIS in $AP_{large,animal}$. The LIS does however degrade the AP in most categories. Specifically, the performance in all large categories and all medium categories, except for $AP_{large,animal}$ seems to be degraded with the use of the LIS.

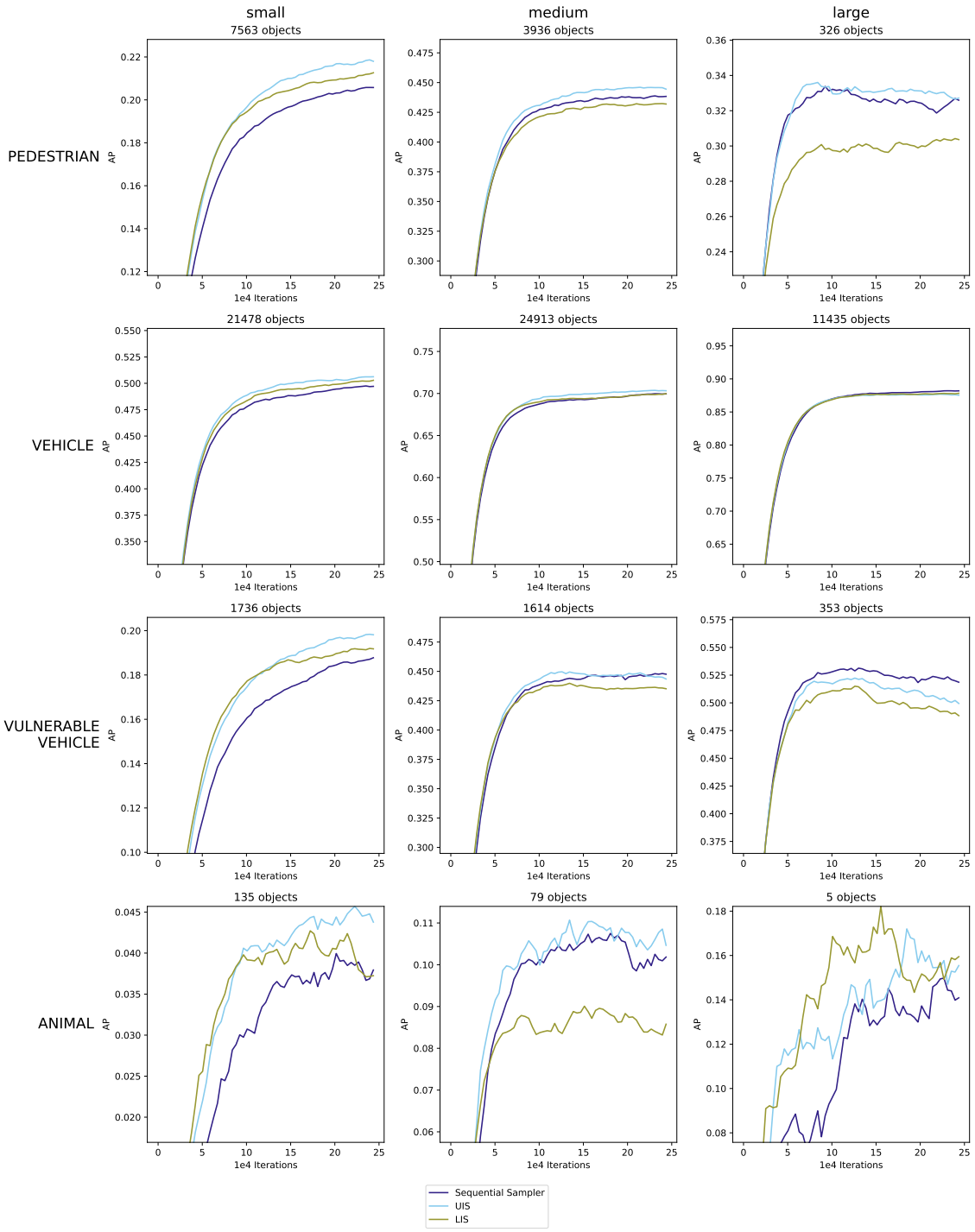

Figure 5.3: Comparison of AP on ZODFAR stratified by bounding box size and class for runs with three different sampling heuristics; Sequential (baseline), LIS, and UIS. Each curve is plotted as an exponential moving average using a smoothing factor of 0.8 for visualization purposes. The corresponding runs can be found unsmoothed in figure A.3 in appendix.

Through closer examination of AP_{animal} and the stratified categories $AP_{small,animal}$, $AP_{medium,animal}$, and $AP_{large,animal}$ (see figure 5.4), it is apparent that the UIS out-

performs both baseline and the LIS on this underrepresented group of data. It is also interesting to note that the LIS underperforms on $AP_{medium,animal}$, and does so with a large enough margin that it degrades its overall AP_{animal}.

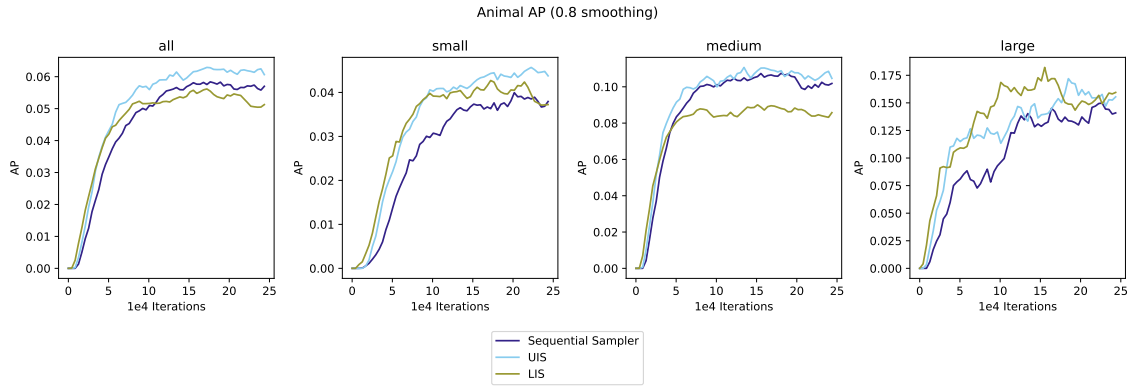

Figure 5.4: Comparison of AP on the underrepresented *Animal* class in ZODFAR for models trained with Sequential (baseline), LIS, and UIS. A smoothing factor of 0.8 is used here for better illustration. The corresponding runs can be found unsmoothed in figure A.4 in appendix.

Interestingly, when evaluating binary AP, that is, detection of the existence of an object rather than also classifying it, we can see that both LIS and UIS outperforms baseline (see figure 5.5).

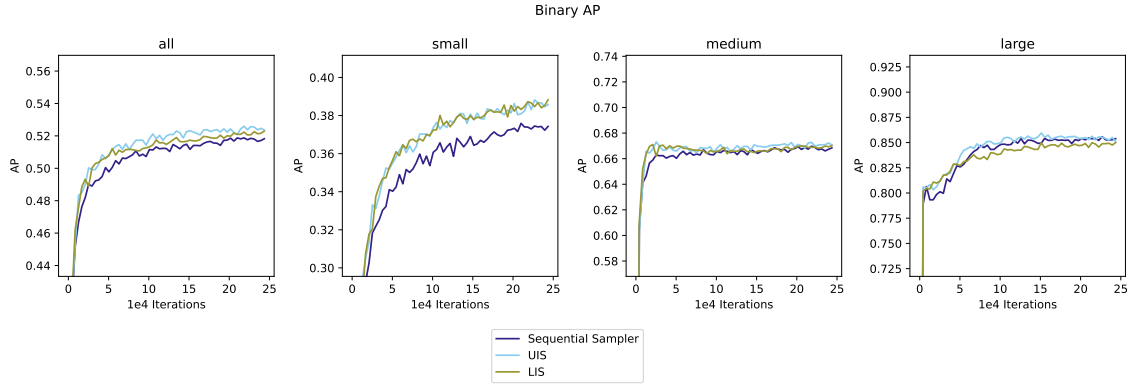

Figure 5.5: Comparison of binary AP between LIS and UIS.

5.2 Rejection-based Sampling Strategy

As highlighted in section 2.3.1, the importance scores retrieved from previous observations may poorly reflect the importance given the current state of the model's parameters. This notion prompted us to investigate an alternative sampling strategy outlined in algorithm 5 that differs from the naive sampling strategy. The idea behind our proposed rejection-based strategy is explained in 4.2.7.

In figure 5.6, it can be observed that when using rejection-based sampling, the Uncertainty Importance Sampler with Rejection-based sampling (UIS-RB) still achieves a speedup, however, it is less than when not using rejection sampling. The Loss Importance Sampler with Rejection-based sampling (LIS-RB) does not achieve a speedup, and in fact, performs significantly worse than the naive sampling scheme.

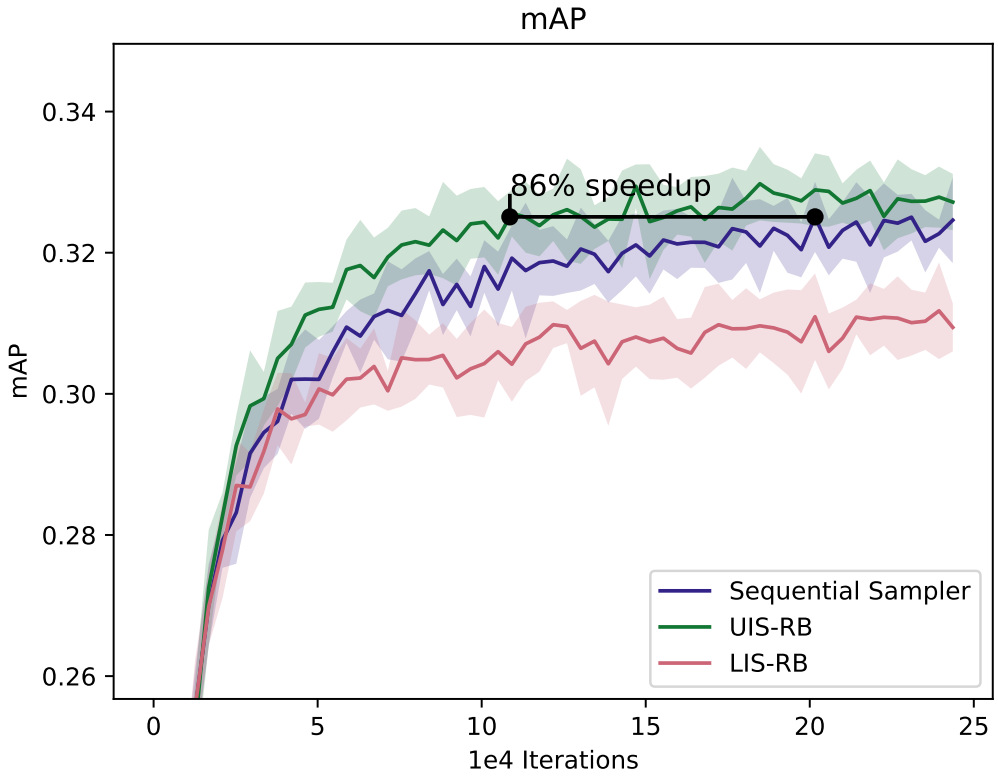

Figure 5.6: Comparison of overall *mAP* on ZODFAR for models trained with Sequential (baseline), LIS-RB, and UIS-RB. The closed black line marks the speedup for achieving the baseline's highest averaged mAP. The filled-in area marks ±1 standard deviation.

Based on the size stratified mAP for the rejection-based sampler's runs in figure 5.7 in the appendix, the performance degradation in comparison to the sampler's naive sampling counterpart and the baseline becomes even more apparent. From 5.8 it is interesting to note how the LIS-RB degrades performance on all categories except for the majority group medium vehicle.

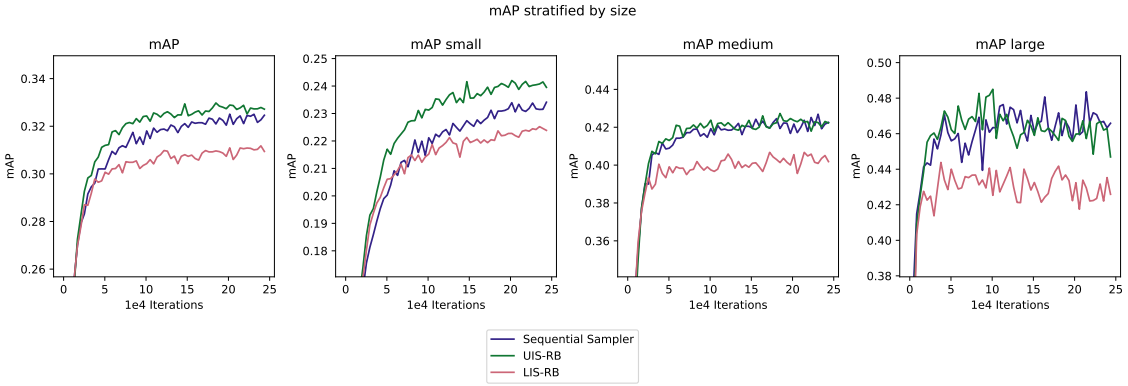

Figure 5.7: Comparison of mAP, mAP_{small}, mAP_{medium}, and mAP_{large} on ZODFAR for models trained with Sequential (baseline), LIS-RB, and UIS-RB.

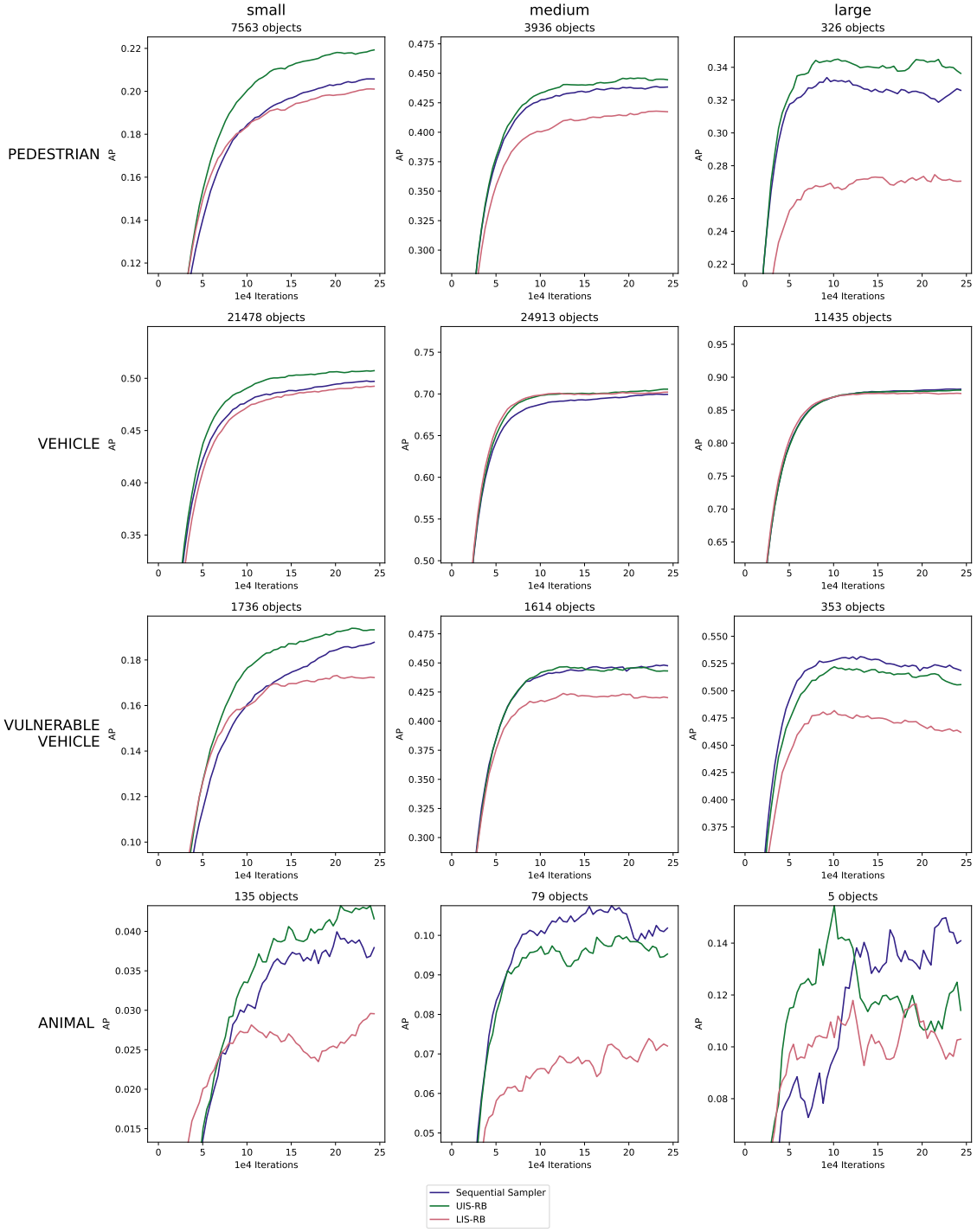

Figure 5.8: Comparison of AP on ZODFAR stratified by bounding box size and class for runs with three different sampling heuristics; Sequential (baseline), LIS-RB, and UIS-RB. Each curve is plotted as an exponential moving average using a smoothing factor of 0.8 for visualization purposes. The corresponding runs can be found unsmoothed in figure A.3 in appendix.

Even the AP on the animal class suffers from poor performance with the rejection-based approach. Surprisingly, the UIS-RB shows subpar performance on this group

of data, which stands in contrast to the performance showcased by its naive sampling counterpart UIS.

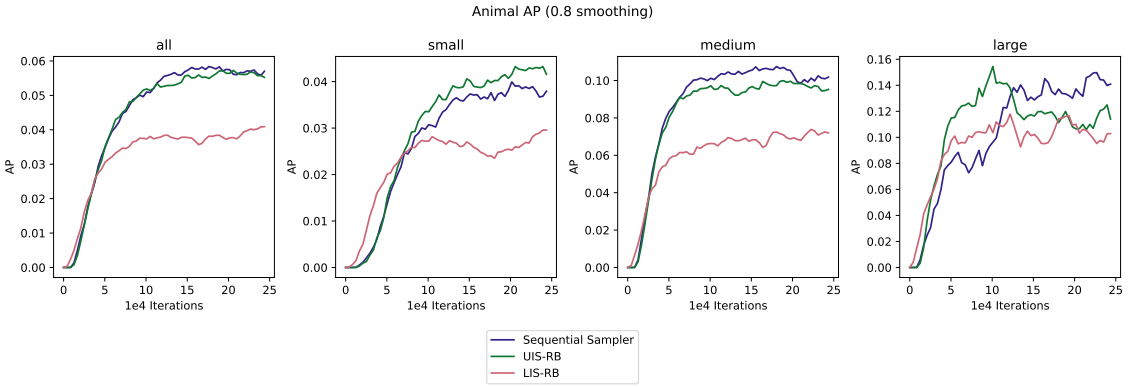

Figure 5.9: Comparison of AP on the underrepresented *Animal* class in ZODFAR for models trained with Sequential (baseline), LIS-RB, and UIS-RB. A smoothing factor of 0.8 is used here for better illustration. The corresponding runs can be found unsmoothed in figure A.4 in appendix.

The degraded performance following the introduction of rejection-based sampling on LIS and UIS is also reflected when evaluating the binary AP (see figure 5.10). Most noticeable is the performance degradation to LIS-RB which, although not as prominent as the LIS, still shows a slight edge over the baseline when it comes to speedup and generalization performance.

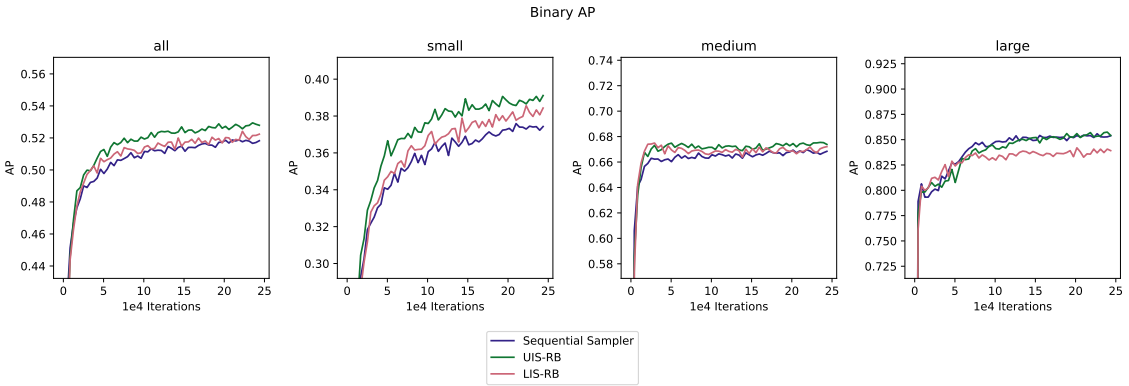

Figure 5.10: Comparison of binary AP between LIS-RB and UIS-RB.

5.3 Reducible Hold-Out loss

Inspired by the claims of prominent speed-up by Mindermann *et al.* [9] and based on the sub-optimal performance resulting from the LIS, the RHO framework was explored as a means to resolve any implicit problems that might have limited the sampler. For instance, from the claims of Chang *et al.* [11] and inferred by the average final layer margin produced by the samples under the LIS's influence (see figure

6.2 in Discussion), it could be reasonable to suspect some level of over-exploitation of noisy, overly ambiguous and out-of-distribution samples [39]. This suspicion and resulting assumptions are discussed in more detail in section 6.1.1.

Similarly to rejection-based sampling, when we apply RHO, the performance becomes degraded in comparison to the baseline. In fact, as conveyed by figure A.1 in the appendix, it is apparent that Loss Importance Sampler with Reducible Hold Out framework (LIS-RHO) is among the least performant types of sampling strategies.

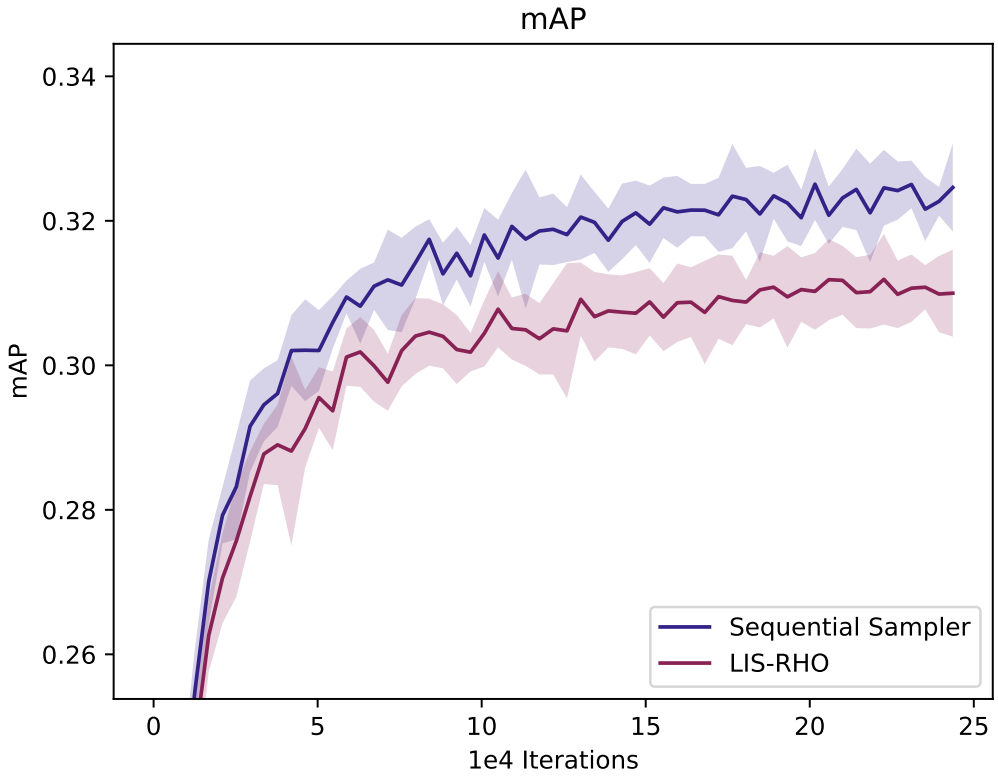

Figure 5.11: Comparison of overall *mAP* on ZODFAR for models trained with Sequential (baseline) and LIS-RHO. The closed black line marks the speedup for achieving the baseline's highest averaged mAP. The filled-in area marks ±1 standard deviation.

Following the trail of LIS-RB, LIS-RHO shows to underperform on all size stratified mAPs (see figure 5.12). While it performs on par or slightly better than the baseline on the majority class vehicle, it significantly sacrifices performance on the remaining categories (see figure 5.13). In particular, the minority data groups, where judging by figure 5.14 and figure 5.13, the *Animal* class seem to suffer the most.

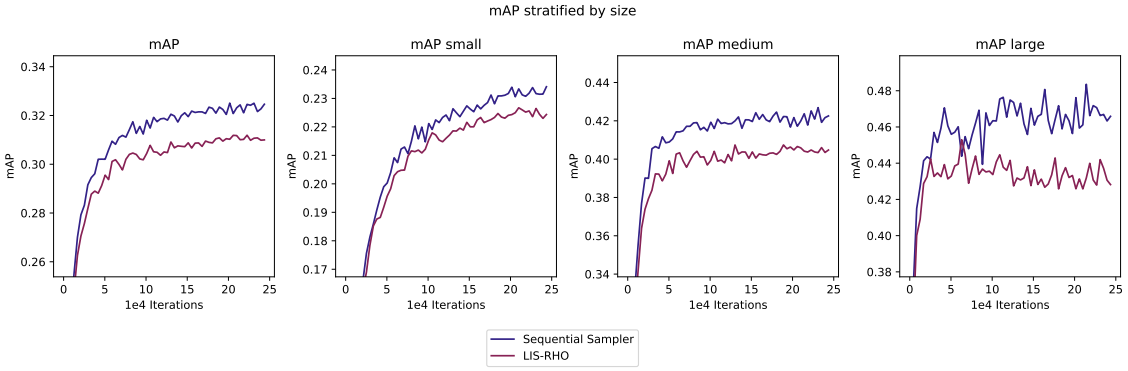

Figure 5.12: Comparison of mAP, mAP_{small}, mAP_{medium}, and mAP_{large} on ZOD-FAR for models trained with Sequential (baseline) and LIS-RHO.

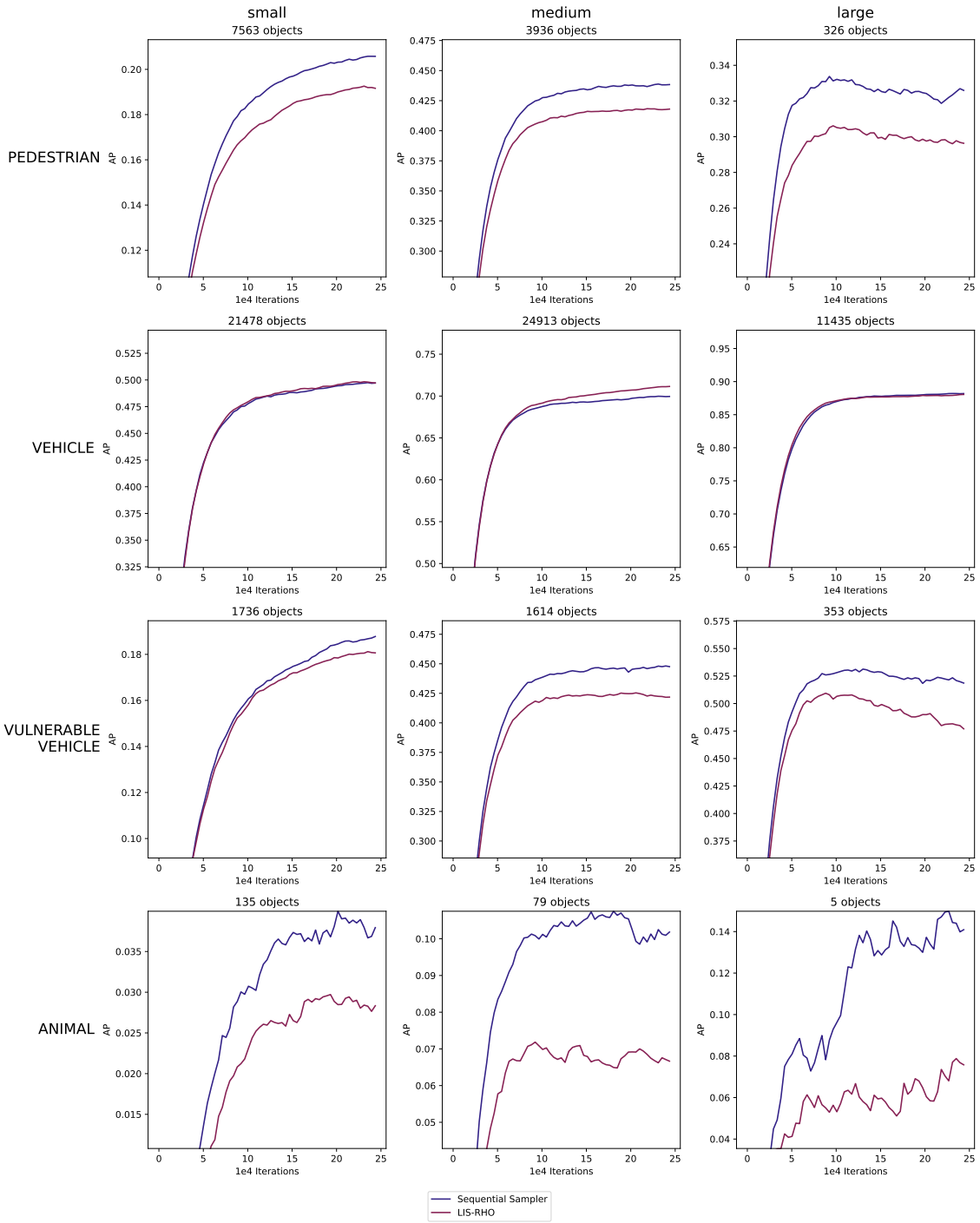

Figure 5.13: Comparison of AP on ZODFAR stratified by bounding box size and class for runs with three different sampling heuristics; Sequential (baseline) and LIS-RHO. Each curve is plotted as an exponential moving average using a smoothing factor of 0.8 for visualization purposes. The corresponding runs can be found unsmoothed in figure A.3 in appendix.

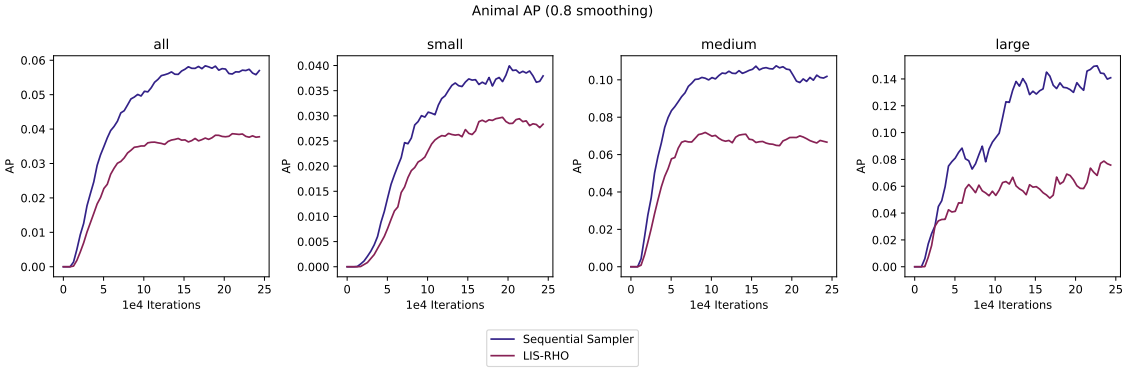

Figure 5.14: Comparison of AP on the underrepresented *Animal* class in ZODFAR for models trained with Sequential (baseline) and LIS-RHO. A smoothing factor of 0.8 is used here for better illustration. The corresponding runs can be found unsmoothed in figure A.4 in appendix.

Although LIS-RHO degrades performance on numerous metrics, when it comes to binary AP, it is actually on par with the baseline as can be seen in figure 5.15.

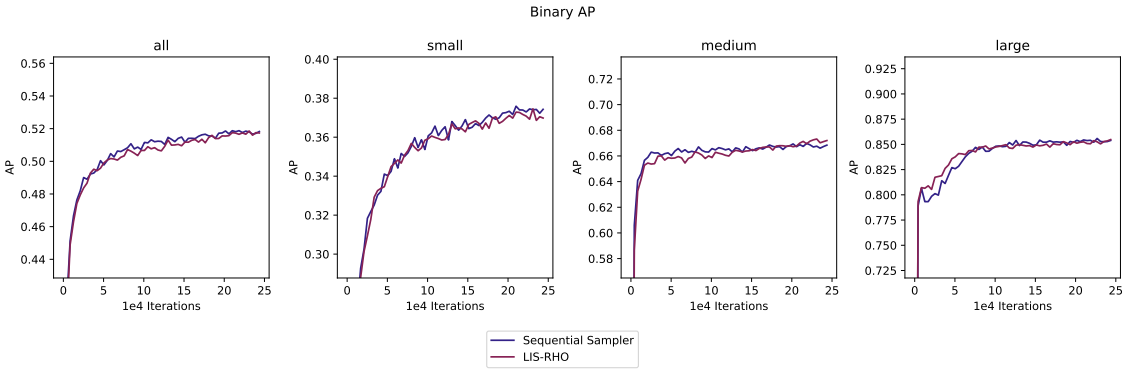

Figure 5.15: Binary AP for LIS-RHO.

5.4 Regression-agnostic Importance Sampler

As indicated by figure 5.5, both LIS and UIS seem to perform better than baseline at detecting objects when evaluating it as a binary task. That is, ignoring the object class and only predicting the existence of objects. This is especially interesting considering the subpar performance shown by the LIS in the other experiments. It may even suggest that the LIS can, from the outset, improve the model's ability to detect objects' presence, but does not facilitate its ability to distinguish their class. Based on this, and the observation that the performance gains seem to be biased towards smaller objects for both samplers, we felt inclined to investigate the effects of IS when excluding regression from the importance scoring calculations. The method is described in section 4.3 and the runs are denoted as Loss Importance Sampler with Ignored Regression (LIS-IR) and Uncertainty Importance Sampler with Ignored Regression (UIS-IR).

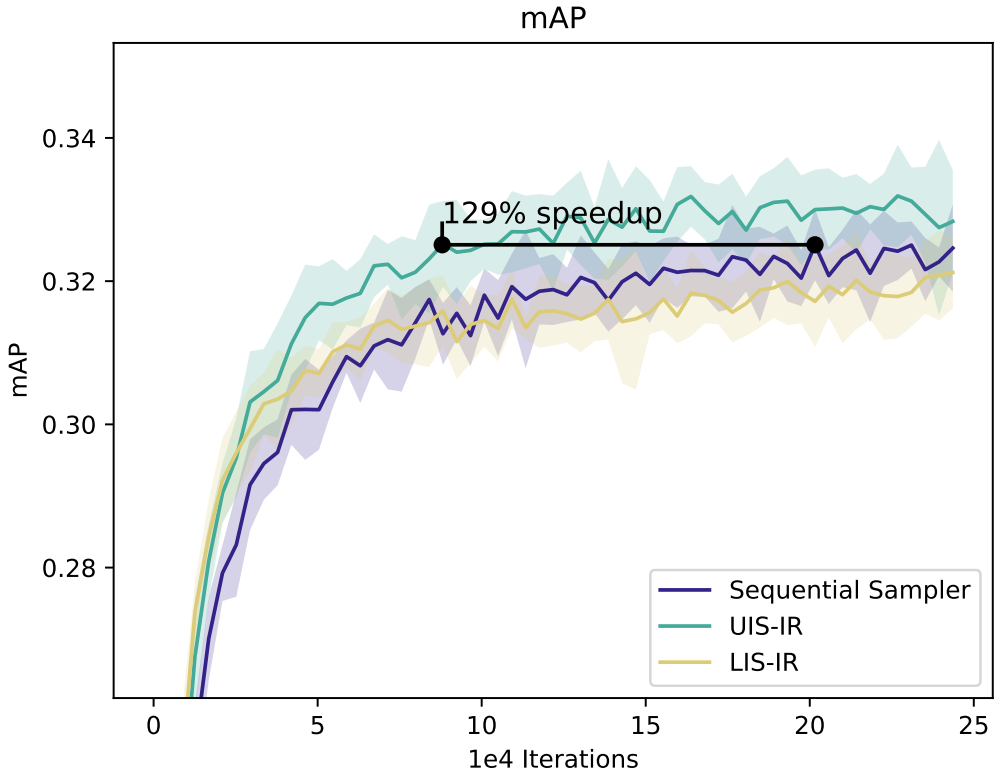

Figure 5.16: Comparison of overall *mAP* on ZODFAR for models trained with Sequential (baseline), Loss Importance Sampler with Ignored Regression (LIS-IR), and UIS-IR. The closed black line marks the speedup for achieving the baseline's highest averaged mAP. The filled-in area marks ±1 standard deviation.

The overall mAP in figure 5.16 shows further improvement with the UIS-IR over the UIS, which now reached a speedup of 129% instead of 95%. The LIS-IR did not manage to show any improvement when ignoring the regression in the sampling. Despite UIS-IR increasing the speedup, it is interesting to note how it sacrifices performance in some areas, in order to gain it in others. When looking at figure 5.17, the UIS-IR performs very well on small objects and sacrifices performance on large objects. In fact, as can be seen in figure A.2 in the appendix, it notably outperforms all other types of runs on small objects.

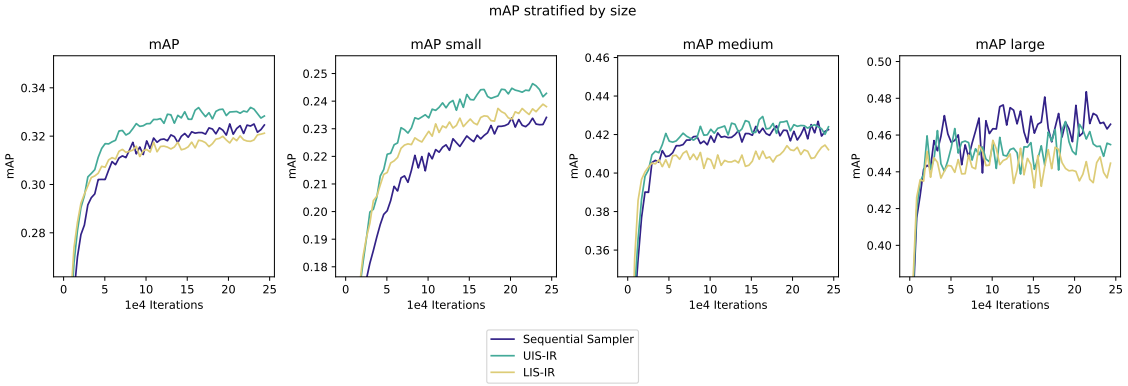

Figure 5.17: Comparison of mAP, mAP_{small}, mAP_{medium}, and mAP_{large} on ZOD-FAR for models trained with Sequential (baseline), LIS-IR, and UIS-IR.

When also looking at class and categories stratified by both class and size, it is evident that UIS-IR is inferior to UIS in this regard. While its overall mAP shows to be better, it does not manage to, unlike its counterpart, achieve speed-up or enhance generalization performance on the *Animal* class across the board (see figure 5.19).

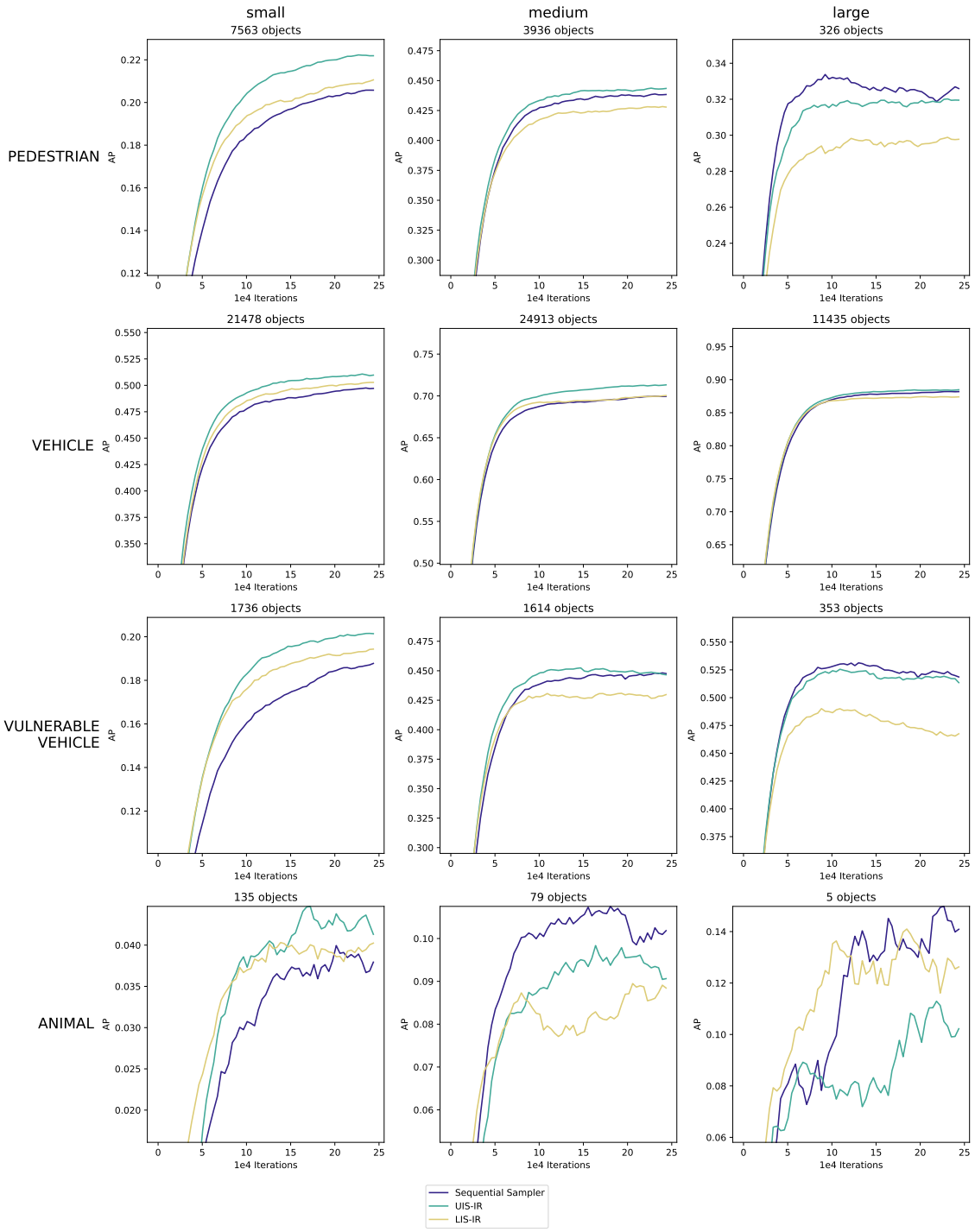

Figure 5.18: Comparison of AP on ZODFAR stratified by bounding box size and class for runs with three different sampling heuristics; Sequential (baseline), LIS-IR, and UIS-IR. Each curve is plotted as an exponential moving average using a smoothing factor of 0.8 for visualization purposes. The corresponding runs can be found unsmoothed in figure A.3 in appendix.

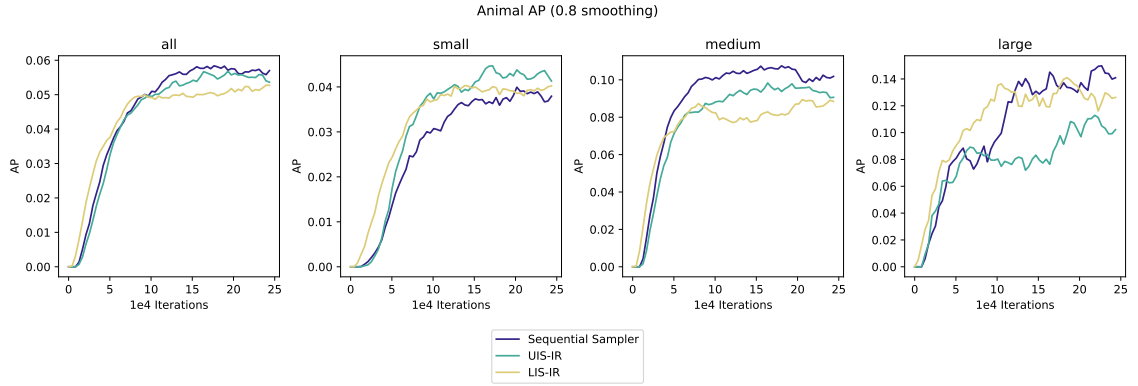

Figure 5.19: Comparison of AP on the underrepresented *Animal* class in ZODFAR for models trained with Sequential (baseline), LIS-IR, and UIS-IR. A smoothing factor of 0.8 is used here for better illustration. The corresponding runs can be found unsmoothed in figure A.4 in appendix.

Similarly to their regular counterpart LIS and UIS, the samplers that ignore regression also outperforms the baseline when it comes to binary AP for all objects as can be seen in figure 5.20. However, while UIS seemed to have a slight edge over the baseline on binary AP for large objects, UIS-IR does not seem to provide any performance increase in this comparison.

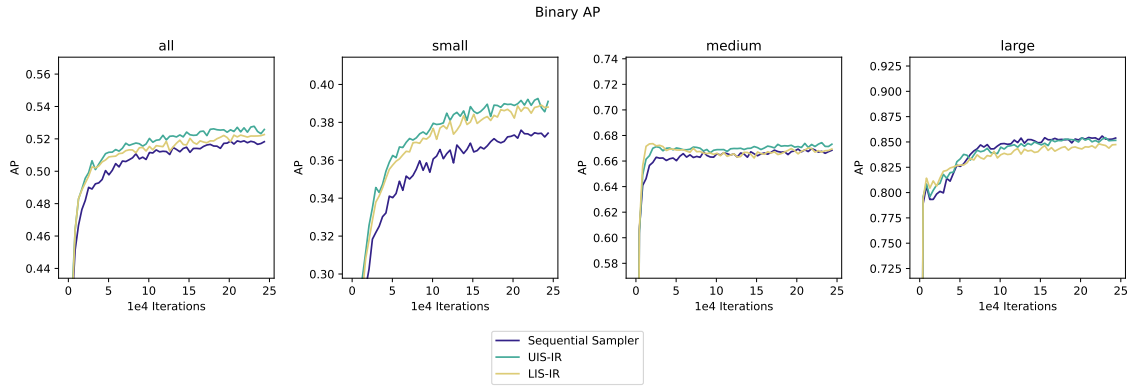

Figure 5.20: Comparison of binary AP between LIS-IR and UIS-IR.

6

Result Analysis

The speedup gained from the use of IS is clear from the results, but interesting details emerge when examining metrics that are not pure performance metrics. This chapter goes into detail about the potential reasons for the reported results and addresses our findings in the context of the research questions stated in section 1.1. Other aspects of IS are also discussed, such as when it might be appropriate to use, if IS is consistent across runs, and if the reported results are reliable.

6.1 Interpretation of the results

A clear superiority is displayed by the UIS and all its variants over the baseline and corresponding LIS runs. This holds true when evaluating the speed-up and final generalization performance on overall mAP, mAP stratified by size, AP on a minority class such as *Animal*, and even binary AP (see figure A.5). The following subsections go into more detail about why this might be the case.

6.1.1 A closer look at loss and uncertainty samplers

The results presented in section 5 suggest that selecting training samples with objects where the model was previously close to the decision boundary is an effective approach. This claim aligns well with Chang *et al.* [11]'s study where samplers that focus on high prediction-variance samples and high threshold-closeness samples outperform samplers that focus on high loss samples. Furthermore, their work demonstrates that while loss-based samplers can be beneficial in less complex tasks, they are found to be sub-optimal compared to uncertainty-based samplers in more complex tasks specifically. This conclusion along with the results reported here are seemingly complementary to the results of Johansson and Lindberg [14]. The authors unveil promising results on a loss-based approach for less complex models on image classification, which does not generalize to the more complex task of OD.

The intuition for leveraging samples for which the model's prediction is uncertain is a phenomenon with supporting claims in self-paced learning. Kumar *et al.* [38] reveal that prioritizing samples that are yet not too difficult for the model (in terms of loss), facilitates learning and avoids overshooting with a smoother, faster, and improved convergence. This effect could, in accordance, explain the success of the UIS. Furthermore, it is interesting to note how the Pearson correlation coefficients

between the UIS's importance scoring and the losses evolve throughout training. From figure 6.1, it can be noted that the UIS begins with a slight negative correlation between classification loss and the importance score, which then increases as training proceeds. This suggests that the sampler starts by concentrating on moderately easy samples and gradually shifts to prioritizing more challenging ones. The same intrinsic mechanism is invoked by curriculum learning, which, although empirically proven to be a technique to speed up training in numerous tasks, requires a substantial amount of preparatory work to be effective.

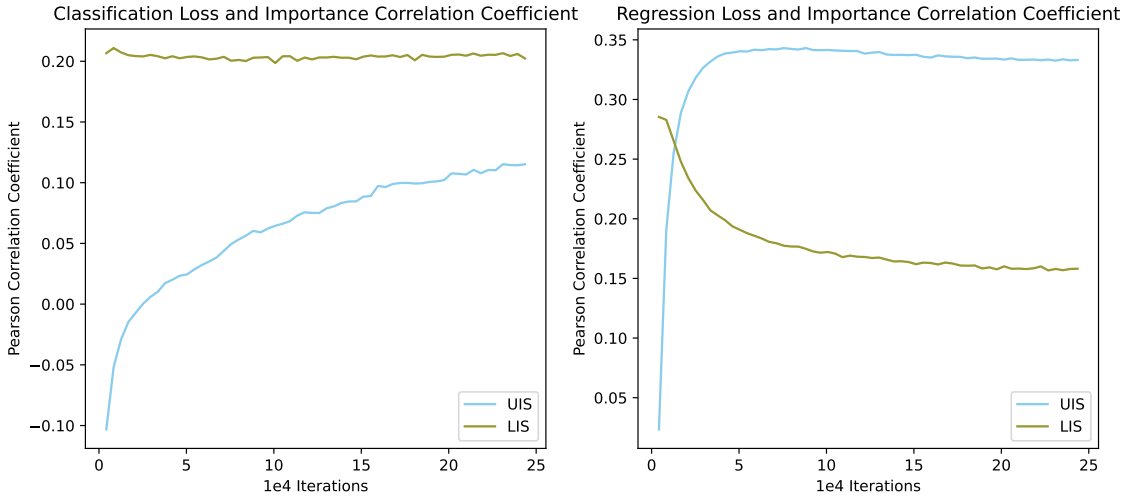

Figure 6.1: Pearson correlation coefficient between classification/regression loss and importance over the course of the training. The values are averaged over 10 runs for each sampler.

As for the LIS, it is not entirely clear why it does not manage to accelerate training or improve overall mAP. However, as indicated by Chang *et al.* [11], and when looking at its average final layer margin of the most frequently drawn samples in figure 6.2, it could be reasonable to suspect some level of over-exploitation of overly ambiguous, noisy, and out-of-distribution samples. Recall that these kinds of samples tend to, in a general case, maintain the most negative final layer margin over the training phase [11]. With this in mind, the top 100 exploited samples for an LIS run and a UIS run were manually evaluated. From this, it was noted that the LIS's top 100 exploited samples consisted of a higher concentration of samples with overly ambiguous and mislabeled objects in comparison to UIS; 30% and 4% respectively. Any prominent exploitation of overly ambiguous, out-of-distribution, and noisy samples could be prohibitive since it may generate gradient estimates that are far from representative of the full gradient. An example of a mislabeled sample and an ambiguous sample can be seen in appendix C; figures C.1 and C.3 respectively.

From the aspect of research question 1) **"How do different types of importance sampling heuristics perform in the task of deep learning object detection"**, the findings in this report suggest that object-reduced uncertainty-based heuristics can accelerate training and are superior to corresponding loss-based heuristics.

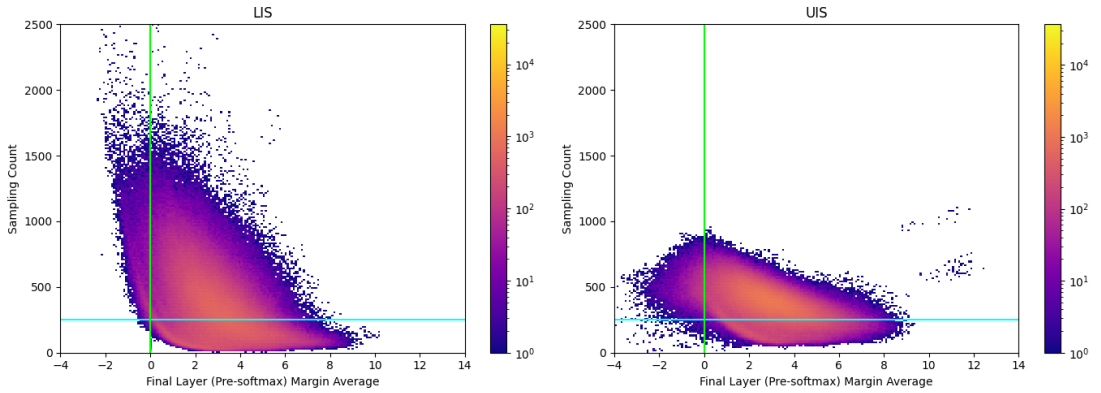

Figure 6.2: 2D histogram of the final layer pre-softmax margin average in the object regions and sampling count. The margin is averaged object-wise at each validation step and then averaged over all the validation steps for each sample. The blue/cyan line marks the number of epochs and can be viewed as a threshold for over/under-sampling. The green line marks the margin average threshold. Samples to the left of the green line tend to have objects that are misclassified during the course of the training. The set of the most negative margin samples within a run could consist of a high concentration of overly ambiguous and mislabeled samples.

6.1.2 RHO Sampling

The results when using RHO show both a slow-down in convergence speed and significantly worse model performance, which stands in contrast to the results shown by Mindermann *et al.* [9]. It is likely that the reason for this is the choice of RHO-specific hyperparameters. The RHO framework has hyperparameters such as the size and architecture of the smaller model, the size of the held-out dataset, how long the smaller model trains, etc. Because of the little time and resources spent toward tuning these hyperparameters, only those presented in table 4.1 were used, so it is possible that the results would have been different with a different choice of hyperparameters.

Another possibility for the degraded performance of the LIS-RHO is that it was not used like Mindermann *et al.* [9] did. They used it in conjunction with online batch selection, where they picked the top-scoring samples to perform backpropagation with. In contrast, this project does not use online batch selection for reasons explained in chapter 3. This discrepancy may be the reason that the results with the RHO sampler were not as promising as they could have been. If it is true that importance sampling with RHO must be done with online batch selection to be effective, it may be the case that RHO is not a suitable framework in data-loading-heavy settings where online batch selection is not appropriate.

6.1.3 Rejection-based Sampling

The rejection-based sampling used in UIS-RB and LIS-RB does not manage to enhance the performance of the regular UIS and LIS. By the looks of figure B.5, B.7,

B.6, and B.8 the samplers become significantly more exploitative and less consistent with this approach. It is likely that the rejection-based algorithm gets stuck on certain samples that elicit the highest importance score. In this case, the model not only over-exploits these samples but does so consecutively, potentially rendering the model more prone to over-fitting. This could be particularly detrimental if it so happens on samples that are harmful to the generalization performance, i.e. overly ambiguous, out-of-distribution, and noisy samples. Connecting to research question 2) **"What are the effects of different sampling strategies when training a deep learning object detector, and how do they perform?"**, it can be clearly stated that a naive sampling strategy can provide a prominent speed-up and final performance increase, while the rejection-based sampling strategy may degrade the performance noticeably.

6.1.4 Ignoring Regression

Ignoring the regression in the importance calculations displayed the highest speedup in the overall mAP. As stated, there were two main reasons for trying this approach: 1) the samplers tend to improve convergence for small objects more than large ones, and 2) the LIS performing better than the baseline in the binary task indicates that classification is the difficult task. Since the effects of ignoring regression were intended to be more class-oriented than size-oriented, it is interesting that the UIS-IR performs better than the regular UIS on small objects but worse on large ones in e.g. the pedestrian class, this can be seen in figure A.3. The LIS-IR, on the other hand, performs worse than the regular LIS consistently, which is the opposite of what was anticipated. In retrospect, it is possible that the reason for this behavior is the don't-care objects. Since these objects do not have a classification score, ignoring the regression means completely ignoring these objects in the importance calculations. This change alone could alter the importance scores of some samples significantly, as it has been empirically observed during the project that many samples containing don't-care objects contain *only* don't-care objects. This would then set the importance for these samples to near-zero, as only the background would be taken into account, and the background is often well predicted. Consequently, the samplers would give the model more objects where there exist gradients that improve the classification, which, as stated, is the more difficult task.

Despite the speed-up attained by UIS-IR, it does not, unlike the UIS, provide a consistent performance enhancement over baseline when it comes to the minority class animal, which stands as a proxy to the low-density regions of the data. As a final note, and to address research question 3) **"Can importance sampling be used to accelerate learning in low-density regions of a data distribution"**: we conclude that the UIS has the potential to accelerate learning in low-density regions of a data distribution.

6.2 Consistency with Importance Sampling

While the results in chapter 5 reveal a possible speedup in mAP, the dynamic and volatile nature of DL training may affect the robustness of these results, especially when a sampling process becomes more erratic by design, and certainly when the average is based on a moderately limited amount of runs. It is therefore important to further question and investigate the consistency of the sampling procedure between runs and samples. To clarify; it may be reasonable to question if the average importance distribution for a specific heuristic and configurations stays consistent between runs, and if for a given sample, its averaged importance is somewhat reflected across multiple runs.

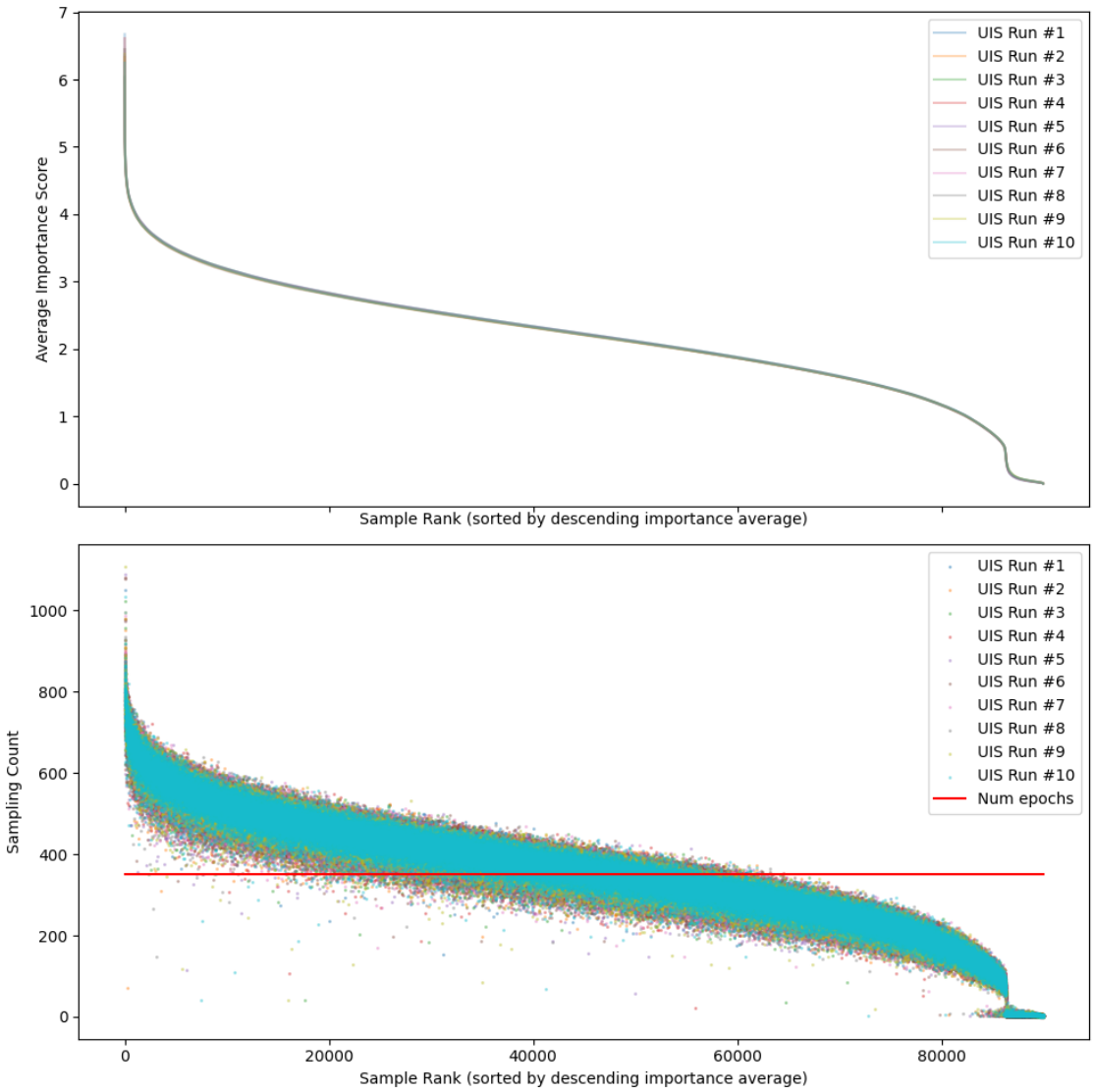

Figure 6.3: Plot of each sample's training averaged importance score for each UIS run (top plot), sorted in descending order, along with their corresponding sample frequency/count during training (bottom plot).

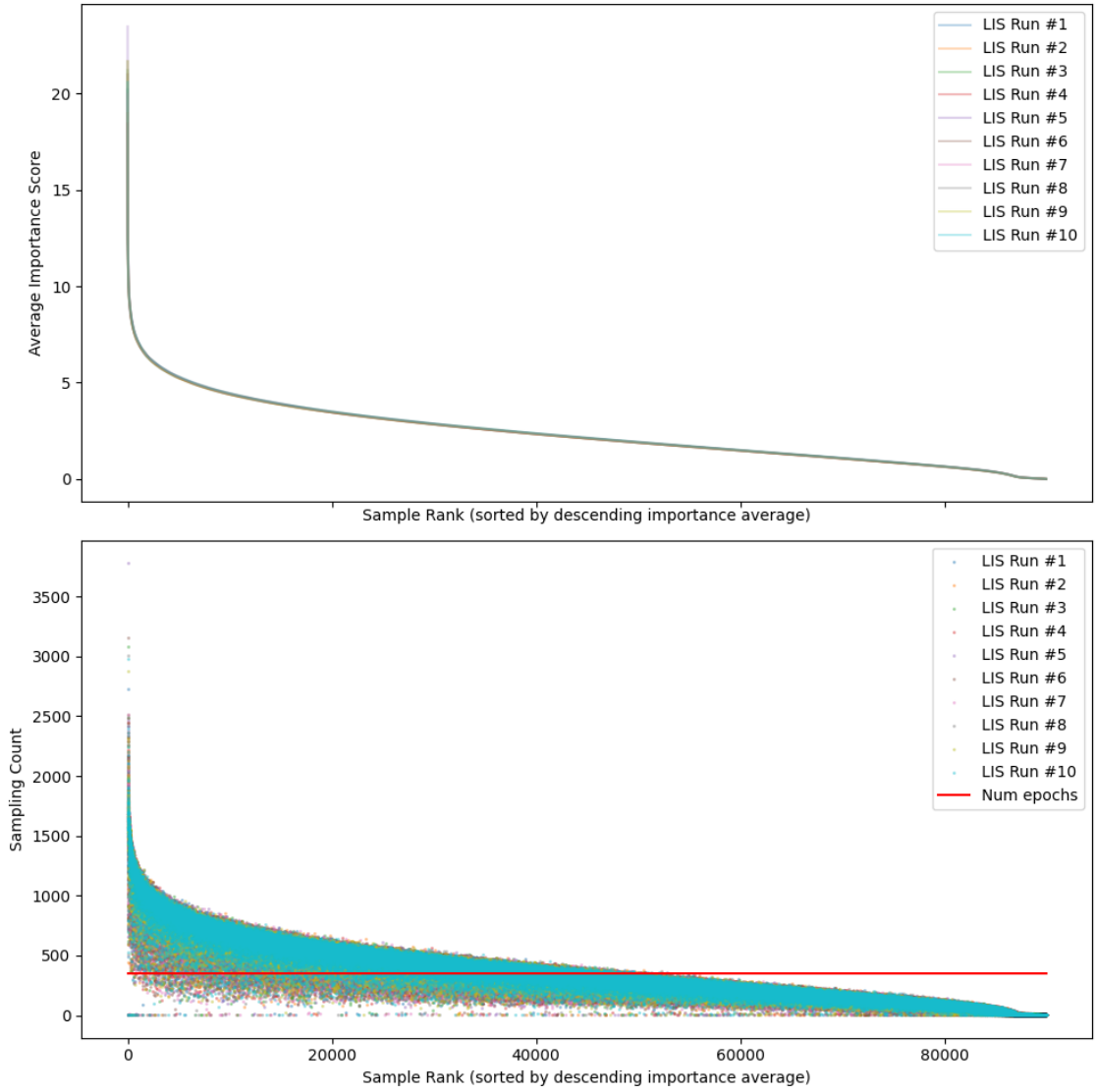

Figure 6.4: Plot of each sample's training averaged importance score for each LIS run (top plot), sorted in descending order, along with their corresponding sample frequency/count during training (bottom plot).

By the look of figures 6.3 and 6.4, it is evident that the average importance and sampling frequency across runs stay highly consistent for both the UIS and the LIS. Unsurprisingly, it can also be noted that the sampling frequency is directly mirrored by the importance average in both cases. The red line in the counts-plots marks the number of epochs and represents the sampling frequency had the sampling been uniform without replacement. Note that each run sorts its samples independently in both figures, and so e.g. the most important sample in one run does not need to be the same sample in another. Figures 6.5 and figure 6.6 indicate a high level of consistency for individual samples across runs, meaning that a sample's average importance score one run tends to be similar in other runs with the same heuristic.

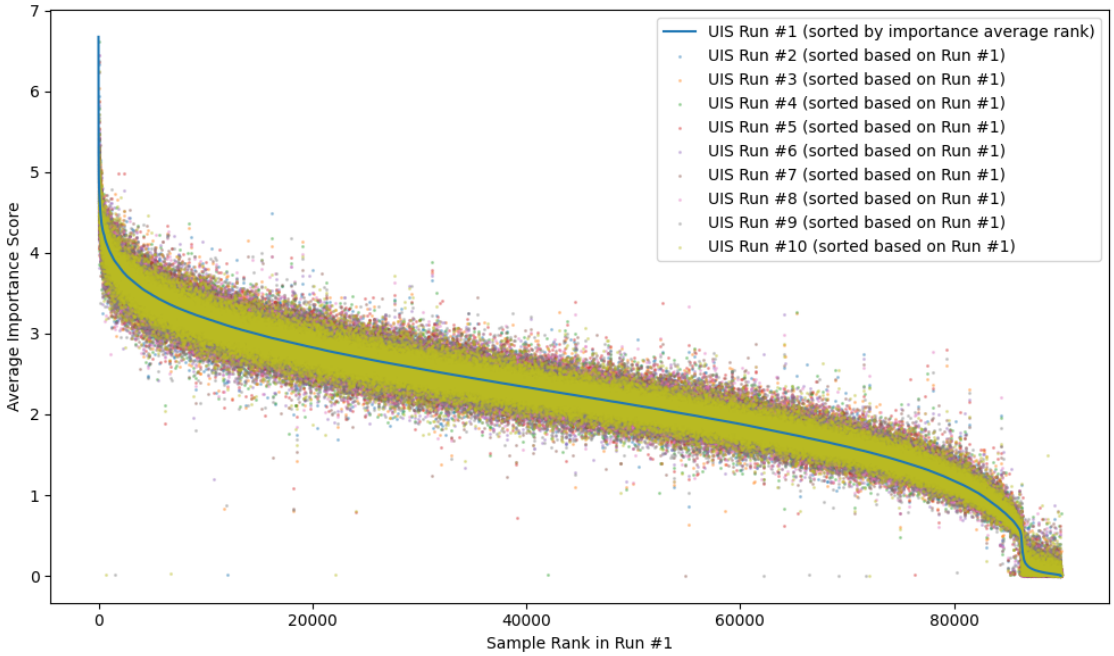

Figure 6.5: Each sample's training averaged importance score across UIS runs, sorted based on a single run.

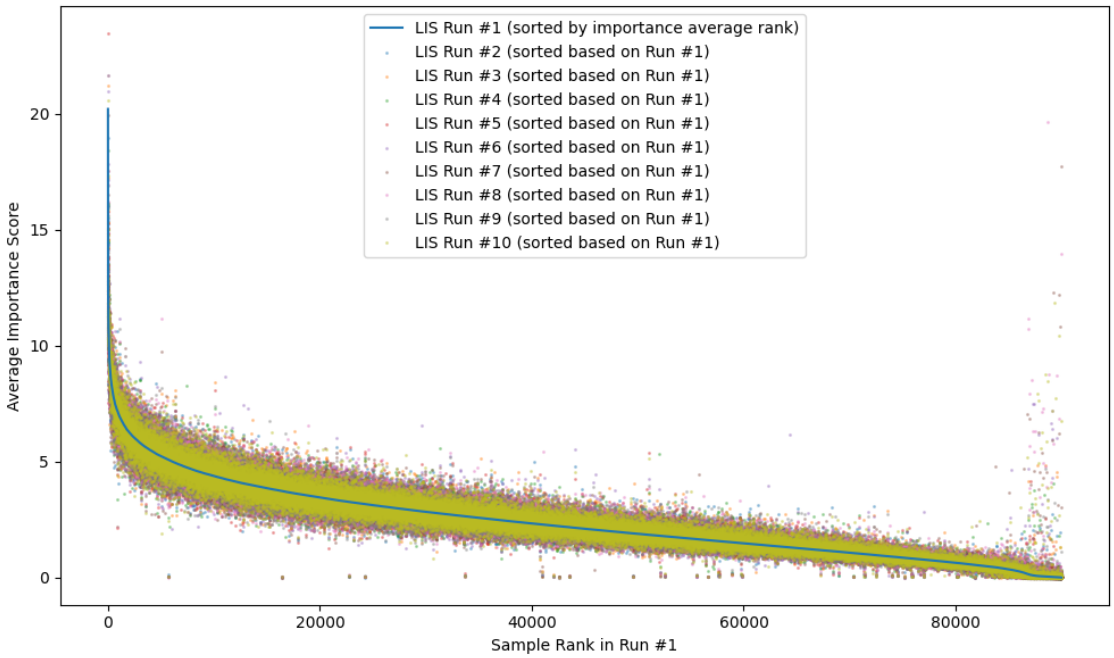

Figure 6.6: Each sample's average importance across LIS runs sorted based on a single run.

6.3 Limitations

The results discussed in the above sections seem promising, but some sacrifices are made. This section details some of the limitations of the methods used in this thesis and some potential pitfalls of importance sampling in general.

6.3.1 Compatibility with Bell and Whistles

Despite the compelling results, it is interesting to note a pattern that is mutual for the majority of IS runs, namely how the samplers improve or speed up performance on small objects specifically. It is unexpectedly the case that the model finds the small objects more challenging, as the baseline performance on these objects is relatively low in comparison to larger ones. However, this problem can be addressed using other techniques, such as object normalization, which scales the pixel-wise loss used for the gradient such that large objects do not dominate the global loss. It remains unclear what the result would have been, had such a technique been combined with the IS approaches presented in this thesis. This is important to emphasize, given that practitioners typically employ various techniques to further enhance the performance of object detectors.

6.3.2 The Background Class and False Positives

The importance sampling approaches examined in this thesis are designed to focus on the performance of the model within the ground truth objects. While this has given positive results, it is worth noting that background is also a class. While the loss used for the gradient always includes the background pixels, ignoring background in the importance sampling can yield many more false positives than using sequential sampling. The object-reduced samplers in this thesis consider all of the background pixels to be one object (only for classification) in order to account for this, but on average, they still end up predicting more false positives than the baseline (see figure 6.7).

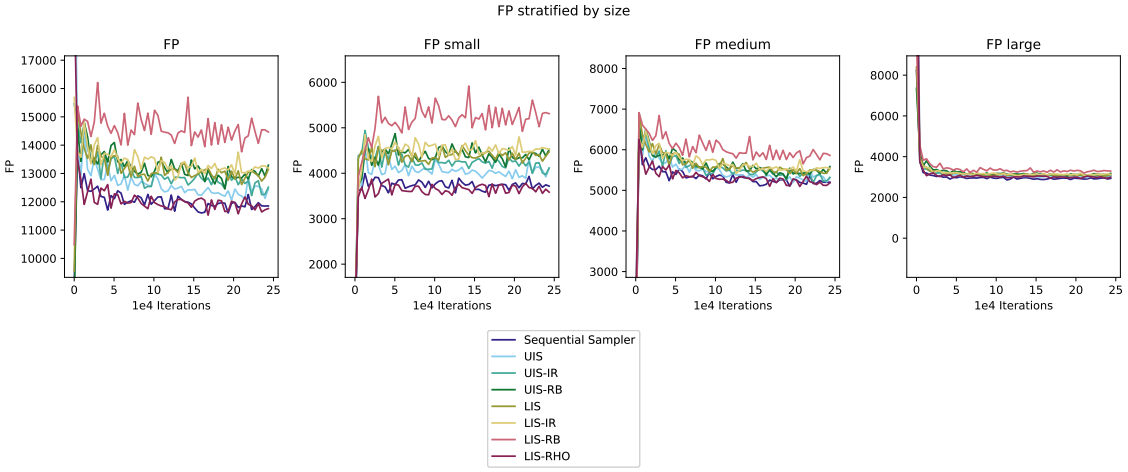

Figure 6.7: Comparison of the number of FPs on ZODFAR over the training course, averaged over multiple runs and stratified by size and class.

Another important note is that the importance score may poorly reflect the sample's global loss. The relatively low correlation coefficients shown in figure 6.1 are an indication of this issue. This may prompt scenarios where a sample could have high importance but negligible loss, hence leading to minuscule gradient contributions.

6.3.3 Sacrificing Performance

The application of importance sampling explicitly re-prioritizes the training data; some samples become more prioritized, and some less. It is then intuitive that if the model trains less on some type of data, it would become worse at predictions for that type of data. We can see this phenomenon in figure 5.3, where the model trained with the UIS is not in fact the best model in all categories. It has, for instance, sacrificed some performance for large vulnerable vehicles.

In the context of AD, one must be very cautious of sacrificing performance in this manner, especially if the performance of a majority class was sacrificed in favor of a minority class. As a hypothetical example, if the AP of Vehicle decreased by 0.01 and the AP of Animal increased by 0.05, the mAP would increase. However, the average driver might drive past hundreds or thousands of other vehicles per day, while not driving past more than a few animals in a month. For this reason, it is of utmost importance that any performance degradation is monitored when using IS.

6.4 Conclusion

This thesis has resulted in promising results for accelerating training in DL OD using IS. This was achieved without the need to tune any hyperparameters or any direct modifications to the loss function. The seemingly best strategy for speed-up and generalization performance in terms of overall mAP is the UIS-IR. The measured speedup using this approach is 129%. When it also comes to performance in low-density regions of the data distribution, the UIS seems to be the most robust choice,

as it was the only type of run that consistently outperformed the baseline on the animal class. Complementary to these results, the observations also indicate that a loss-based heuristic is sub-optimal in DL OD. This holds true even after attempts to amend potential issues using, for instance, the RHO framework or a rejection-based sampling scheme. The above can be stated with confidence since it has also been shown that the sampling strategies are consistent over several runs, as each training sample is given approximately the same importance in the different runs.

7

Future Work

During the project, several intriguing aspects of Importance Sampling (IS) in Object Detection (OD) emerged which could warrant further investigations. In particular, the promising speed-up and final performance displayed by the UIS could be a leadway to more conclusive evidence or theoretical guarantees in future work. Furthermore, due to the symmetry of the UIS calculations around the decision boundaries, it would be interesting to evaluate a similar heuristic that weights the pre-logarithm calculation $f(x) = 1 - 4 \cdot x \cdot (1 - x)$, $x \in [0, 1]$ more towards higher errors while still favoring threshold-closeness. This trick could presumably put emphasis on samples that are uncertain and have a greater contribution to the gradient estimates while still down-weighting the out-of-distribution and noisy samples. While on the topic of importance heuristics, and in accordance with the results showing that ignoring regression could be beneficial, it could also be interesting to conduct more experiments with a weighted sum between the classification and regression in the importance calculations rather than ignoring regression completely.

Furthermore, on the topic of improving our work, and indicated by our method's limitations, we believe that a reduction technique that more profoundly considers locations outside of ground truth objects i.e. background and false positives, could be a relevant area to investigate. For instance, it would be interesting to note the effects when also considering locations in the post-processed predicted bounding boxes. Although it may be computationally prohibitive to use something like NMS in order to compute importance score in predicted regions, if such a trick manages to further accelerate training or improve final performance in terms of iteration count, it could potentially open up an interesting strategy to improve upon computationally.

Additionally, with regards to object-wise reduction, we encourage future experiments with various reduction modes, some of which are listed in section 4.2.4, to potentially induce any problem-specific bias. During development, we did experiment with the mentioned reduction modes. Observing the outcome and as per the time constraints, we chose to solely focus on mean reduction. Interestingly but unsurprisingly, granted the choice of reduction-mode, the performance of the model would behave very differently across different modes. Reducing the object's importance scores using an area-weighted mean would bias the sampler towards samples with larger objects that are challenging. With the root-mean-squared reduction, the sampler would prefer samples with many challenging objects. While the mode could be chosen heuristically, this mechanism could be used to address issues like, for instance, class

or size imbalance.

As a final note, based on the compelling evidence garnered by Shrivastava *et al.* [49] and Cao *et al.* [51], extending our methods further to also consider sampling of images' regions could be an interesting area to investigate. In contrast to how we conducted our experiments, OD inputs are commonly cropped at random during training. Hence, a method that additionally samples these crops could result in a greater speed-up or final performance.

Ultimately, continuing the discourse on the most effective importance sampling method is a compelling subject that we believe will hold significant relevance in the future. This is especially relevant when considering the evolving complexity of problems that are being solved with ML and the ever-increasing capacity of corresponding data-generating processes. Granted this, we anticipate that importance sampling in deep learning object detection offers a promising avenue to the acceleration of advancements in a range of fields. For instance, in AD and ADAS, where it could support in driving fatalities towards zero faster.

Bibliography

[1] M. Toneva, A. Sordoni, R. T. des Combes, A. Trischler, Y. Bengio, and G. J. Gordon, "An empirical study of example forgetting during deep neural network learning," *CoRR*, vol. abs/1812.05159, 2018. arXiv: 1812.05159. [Online]. Available: http://arxiv.org/abs/1812.05159.

[2] O. Sener and S. Savarese, *Active learning for convolutional neural networks: A core-set approach*, 2017. DOI: 10.48550/ARXIV.1708.00489. [Online]. Available: https://arxiv.org/abs/1708.00489.

[3] R. J. N. Baldock, H. Maennel, and B. Neyshabur, "Deep learning through the lens of example difficulty," *CoRR*, vol. abs/2106.09647, 2021. arXiv: 2106.09647. [Online]. Available: https://arxiv.org/abs/2106.09647.

[4] M. Paul, S. Ganguli, and G. K. Dziugaite, *Deep learning on a data diet: Finding important examples early in training*, 2021. DOI: 10.48550/ARXIV.2107.07075. [Online]. Available: https://arxiv.org/abs/2107.07075.

[5] X. Zhou, O. Wu, W. Zhu, and Z. Liang, *Understanding difficulty-based sample weighting with a universal difficulty measure*, 2023. DOI: 10.48550/ARXIV.2301.04850. [Online]. Available: https://arxiv.org/abs/2301.04850.

[6] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," vol. 60, Jun. 2009, p. 6. DOI: 10.1145/1553374.1553380.

[7] A. Katharopoulos and F. Fleuret, "Not all samples are created equal: Deep learning with importance sampling," *CoRR*, vol. abs/1803.00942, 2018. arXiv: 1803.00942. [Online]. Available: http://arxiv.org/abs/1803.00942.

[8] P. Zhao and T. Zhang, "Stochastic optimization with importance sampling for regularized loss minimization," in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, Jul. 2015, pp. 1–9. [Online]. Available: https://proceedings.mlr.press/v37/zhaoa15.html.

[9] S. Mindermann, J. Brauner, M. Razzak, *et al.*, *Prioritized training on points that are learnable, worth learning, and not yet learnt*, 2022. DOI: 10.48550/ARXIV.2206.07137. [Online]. Available: https://arxiv.org/abs/2206.07137.

[10] P. Zhao and T. Zhang, *Stochastic optimization with importance sampling*, 2014. DOI: 10.48550/ARXIV.1401.2753. [Online]. Available: https://arxiv.org/abs/1401.2753.

[11] H.-S. Chang, E. Learned-Miller, and A. McCallum, *Active bias: Training more accurate neural networks by emphasizing high variance samples*, 2017. DOI:

10.48550/ARXIV.1704.07433. [Online]. Available: https://arxiv.org/abs/1704.07433.

[12] I. Loshchilov and F. Hutter, "Online batch selection for faster training of neural networks," *CoRR*, vol. abs/1511.06343, 2015. arXiv: 1511.06343. [Online]. Available: http://arxiv.org/abs/1511.06343.

[13] Zenseact, *About*, https://zenseact.com/people-at-heart/, Accessed April 20, 2023.

[14] M. Johansson and E. Lindberg, *Importance sampling in deep learning*, 2022. [Online]. Available: http://uu.diva-portal.org/smash/get/diva2:1685645/FULLTEXT01.pdf.

[15] Y. Mo, Y. Wu, X. Yang, F. Liu, and Y. Liao, "Review the state-of-the-art technologies of semantic segmentation based on deep learning," *Neurocomputing*, vol. 493, pp. 626–646, 2022, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2022.01.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222000054.

[16] A. Byerly, T. Kalganova, and R. Ott, "The current state ofătheăart inădeep learning forăimage classification: A review," in *Intelligent Computing*, K. Arai, Ed., Cham: Springer International Publishing, 2022, pp. 88–105, ISBN: 978-3-031-10464-0.

[17] S. A. J. Naqvi and S. B. Ali, *State-of-the-art models for object detection in various fields of application*, 2022. DOI: 10.48550/ARXIV.2211.00733. [Online]. Available: https://arxiv.org/abs/2211.00733.

[18] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. DOI: 10.48550/ARXIV.1412.6980. [Online]. Available: https://arxiv.org/abs/1412.6980.

[19] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning*, S. Dasgupta and D. McAllester, Eds., ser. Proceedings of Machine Learning Research, vol. 28, Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1139–1147. [Online]. Available: https://proceedings.mlr.press/v28/sutskever13.html.

[20] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, 2015. arXiv: 1502.03167 [cs.LG].

[21] C. Cortes, M. Mohri, and A. Rostamizadeh, *L2 regularization for learning kernels*, 2012. arXiv: 1205.2653 [cs.LG].

[22] G. Ioannou, T. Tagaris, and A. Stafylopatis, "Improving the convergence speed of deep neural networks with biased sampling," in *Proceedings of the 3rd International Conference on Advances in Artificial Intelligence*, ser. ICAAI '19, Istanbul, Turkey: Association for Computing Machinery, 2020, pp. 35–41, ISBN: 9781450372534. DOI: 10.1145/3369114.3369116. [Online]. Available: https://doi.org/10.1145/3369114.3369116.

[23] E. C. Anderson, "Monte carlo methods and importance sampling," Oct. 1999. [Online]. Available: https://ib.berkeley.edu/labs/slatkin/eriq/classes/guest_lect/mc_lecture_notes.pdf.

[24] R. Y. Rubinstein and D. P. Kroese, "Simulation and the monte carlo method," in John Wiley & Sons, 2016, ch. 5.7, pp. 149–154.

[25] A. Jain, J. Mao, and K. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996. DOI: 10.1109/2.485891.

[26] H. B. Curry, "The method of steepest descent for non-linear minimization problems," *Quarterly of Applied Mathematics*, vol. 2, no. 3, pp. 258–261, 1944.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016, ch. 6.5 Back-Propagation and other Differentiation Algorithms, pp. 200–220.

[28] Y. LeCun, Y. Bengio, and G. Hinton, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*, vol. 3361, Cambridge, MA: MIT Press, 1998, pp. 3361–3366.

[29] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, *Object detection with deep learning: A review*, 2019. arXiv: 1807.05511 [cs.CV].

[30] *Zenseact open dataset*, Accsessed: 2023-03-15. [Online]. Available: https://zod.zenseact.com/.

[31] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020, pp. 237–242.

[32] *Pascal visual object classes*, Accessed: 2023-03-15. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/.

[33] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2017. arXiv: 1706.03762 [cs.CL].

[34] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, *End-to-end object detection with transformers*, 2020. arXiv: 2005.12872 [cs.CV].

[35] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," 2019. arXiv: 1904.01355. [Online]. Available: https://arxiv.org/abs/1904.01355.

[36] W. Lamorte, *Central limit theorem*, 2016. [Online]. Available: https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_probability/BS704_Probability12.html.

[37] G. Hacohen and D. Weinshall, *On the power of curriculum learning in training deep networks*, 2019. arXiv: 1904.03626 [cs.LG].

[38] M. Kumar, B. Packer, and D. Koller, "Self-paced learning for latent variable models," in *Advances in Neural Information Processing Systems*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., vol. 23, Curran Associates, Inc., 2010. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2010/file/e57c6b956a6521b28495f2886ca0977a-Paper.pdf.

[39] G. Pleiss, T. Zhang, E. R. Elenberg, and K. Q. Weinberger, *Identifying mislabeled data using the area under the margin ranking*, 2020. arXiv: 2001.10528 [cs.LG].

[40] C. G. Northcutt, T. Wu, and I. L. Chuang, *Learning with confident examples: Rank pruning for robust classification with noisy labels*, 2017. arXiv: 1705.01936 [stat.ML].

[41] C. G. Northcutt, L. Jiang, and I. L. Chuang, *Confident learning: Estimating uncertainty in dataset labels*, 2022. arXiv: 1911.00068 [stat.ML].

[42] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," in *Advances in Neural Information Processing Systems*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., vol. 26, Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2013/file/3871bd64012152bfb53fdf04b401193f-Paper.pdf.

[43] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness, *How important is importance sampling for deep budgeted training?* 2021. DOI: 10.48550/ARXIV.2110.14283. [Online]. Available: https://arxiv.org/abs/2110.14283.

[44] A. Krizhevsky, "Learning multiple layers of features from tiny images," pp. 32–33, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[45] A. Katharopoulos and F. Fleuret, *Biased importance sampling for deep neural network training*, 2017. arXiv: 1706.00043 [cs.LG].

[46] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[47] A. H. Jiang, D. L. Wong, G. Zhou, *et al.*, "Accelerating deep learning by focusing on the biggest losers," *CoRR*, vol. abs/1910.00762, 2019. arXiv: 1910.00762. [Online]. Available: http://arxiv.org/abs/1910.00762.

[48] L. Tang, A. Jiang, and G. Ganger, *Memoization to reduce forward-passes in selective-backprop*. [Online]. Available: https://www.andrew.cmu.edu/user/liliat/files/report.pdf.

[49] A. Shrivastava, A. Gupta, and R. Girshick, *Training region-based object detectors with online hard example mining*, 2016. arXiv: 1604.03540 [cs.CV].

[50] H. Yu, Z. Zhang, Z. Qin, *et al.*, *Loss-rank-mining-a-general-hard-example-mining-method-for-real-time-detectors*, 2018. arXiv: 1804.04606 [cs.CV].

[51] Y. Cao, K. Chen, C. C. Loy, and D. Lin, *Prime sample attention in object detection*, 2019. arXiv: 1904.04821 [cs.CV].

[52] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV].

[53] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, *Deep layer aggregation*, 2019. arXiv: 1707.06484 [cs.CV].

[54] M. Alibeigi, W. Ljungbergh, A. Tonderski, *et al.*, *Zenseact open dataset: A large-scale and diverse multimodal dataset for autonomous driving*, 2023. arXiv: 2305.02008 [cs.CV].

[55] T. Lin, M. Maire, S. J. Belongie, *et al.*, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. arXiv: 1405.0312. [Online]. Available: http://arxiv.org/abs/1405.0312.

A

Appendix 1

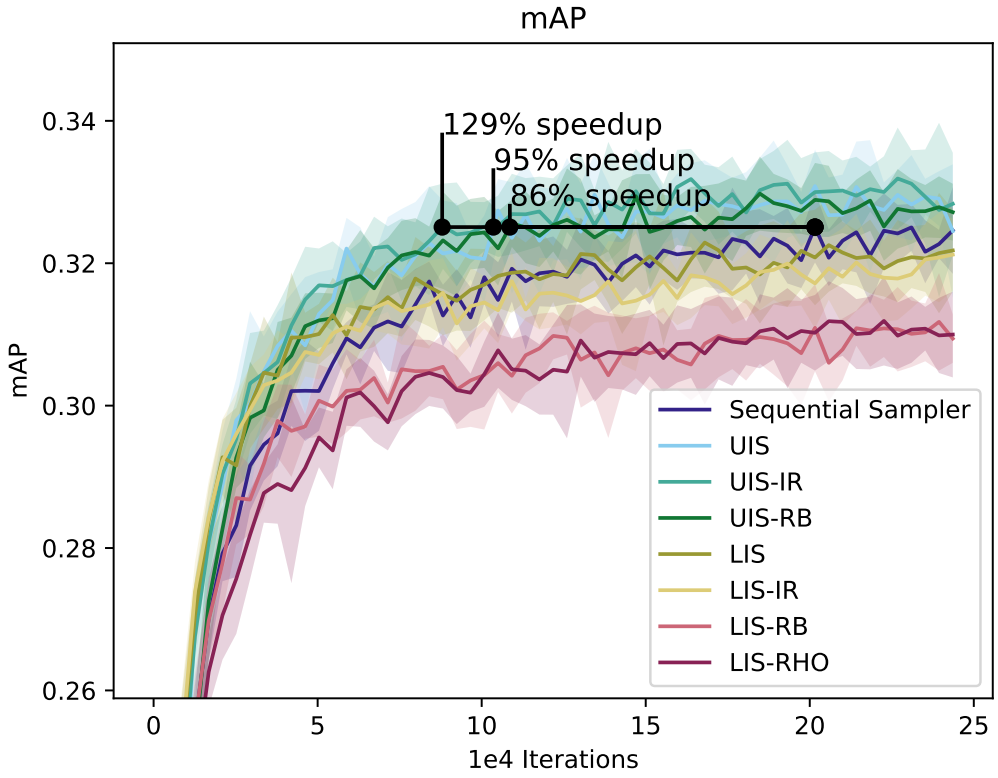

Figure A.1: Comparison of overall mAP on ZODFAR for all types of runs. The closed black line marks the speedup for achieving the baseline's highest averaged mAP.

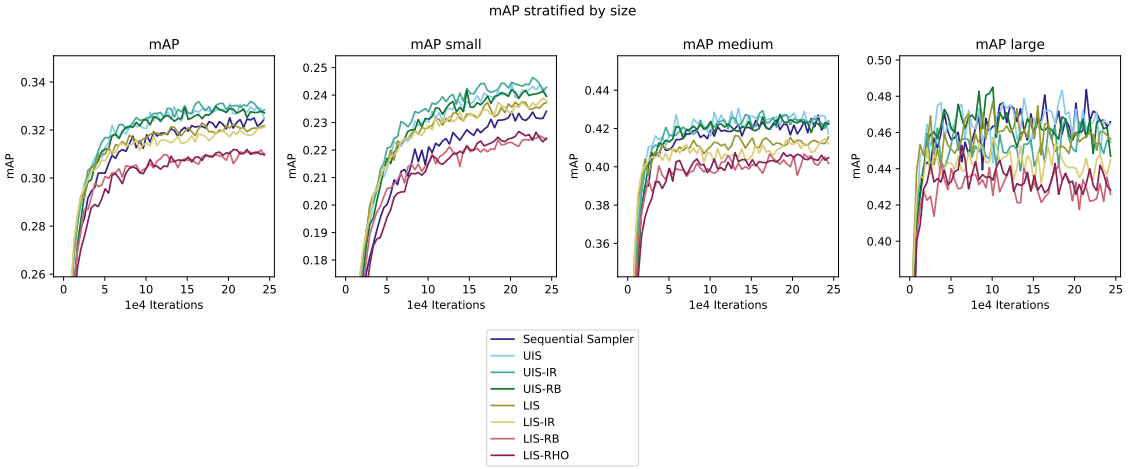

Figure A.2: Comparison mAP stratified by size on ZODFAR for all types of runs.

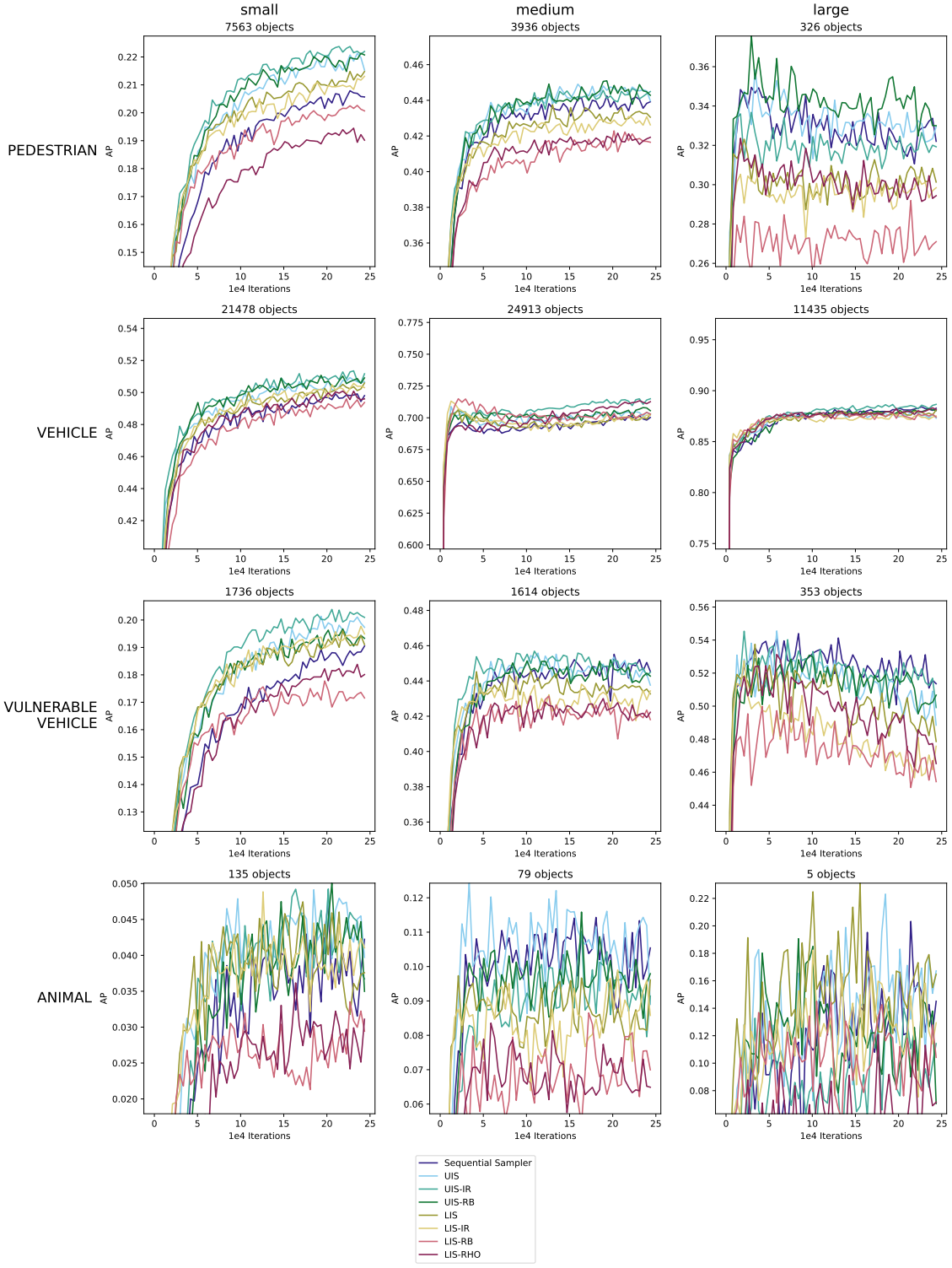

Figure A.3: Comparison of mAP stratified by both size and class on ZODFAR for all types of runs.

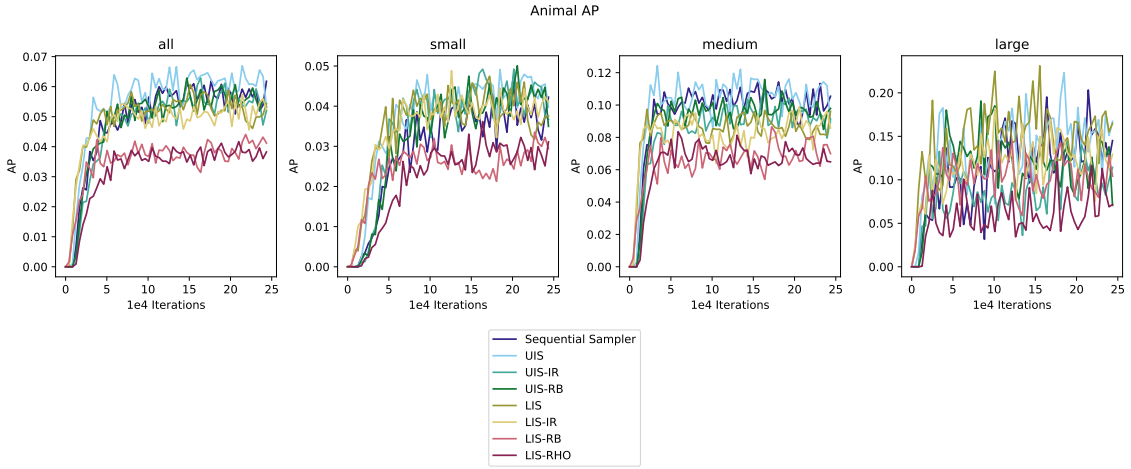

Figure A.4: Comparison of AP_{Animal} stratified by size on ZODFAR for all types of runs.

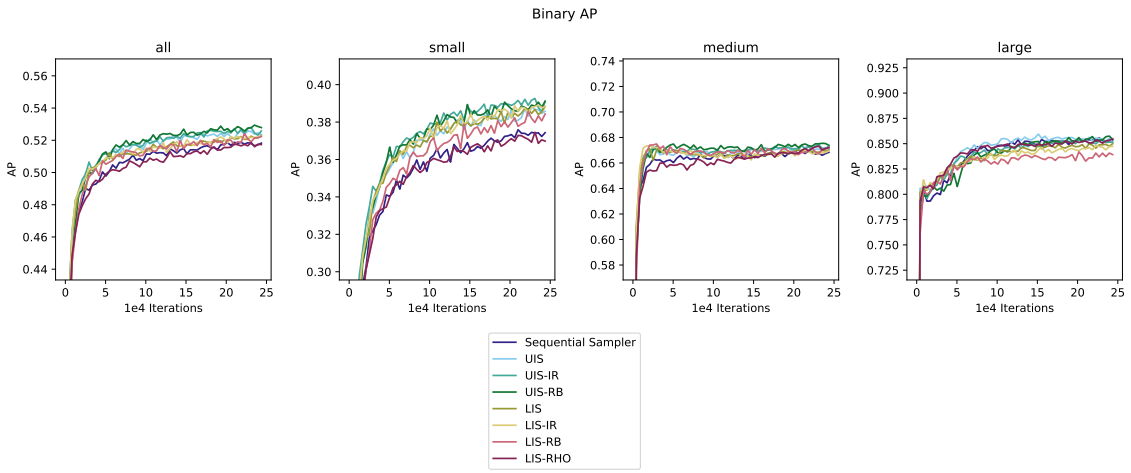

Figure A.5: Comparison of binary AP for all types of runs.

B

Appendix 2

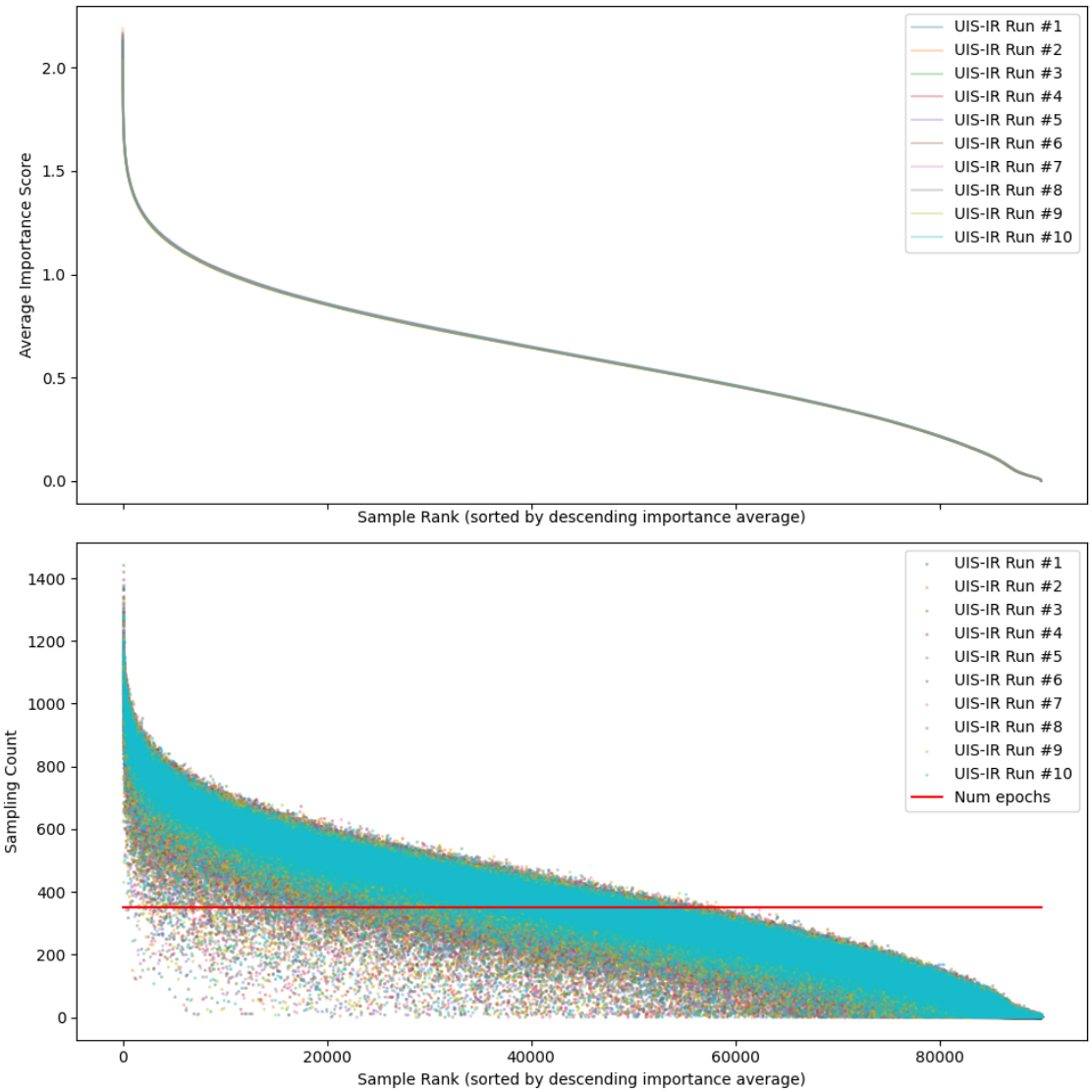

Figure B.1: Plot of each sample's training averaged importance score for each UIS-IR run (top plot), sorted in descending order, along with their corresponding sample frequency/count during training (bottom plot). For clarification; each x position corresponds to the sample in both subplots. The top plot is sorted by each sample's importance intra-run average. The bottom plot shows the corresponding sampling count i.e. the number of times it was sampled during training.

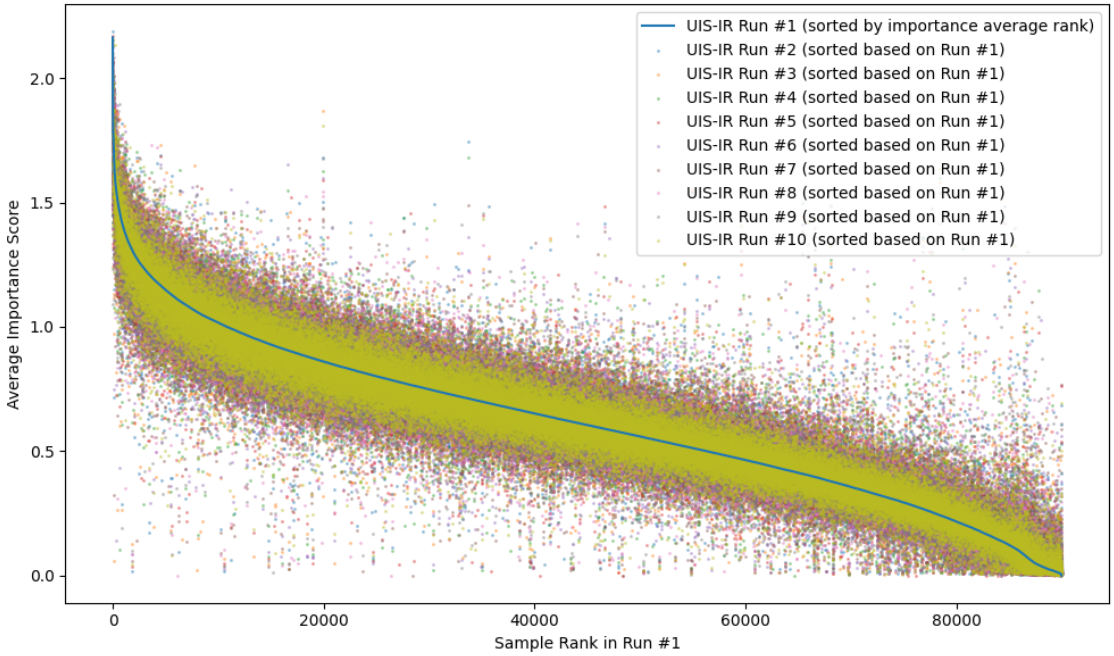

Figure B.2: Each sample's training averaged importance score across runs with (regression ignored) UIS sorted based on a single run.

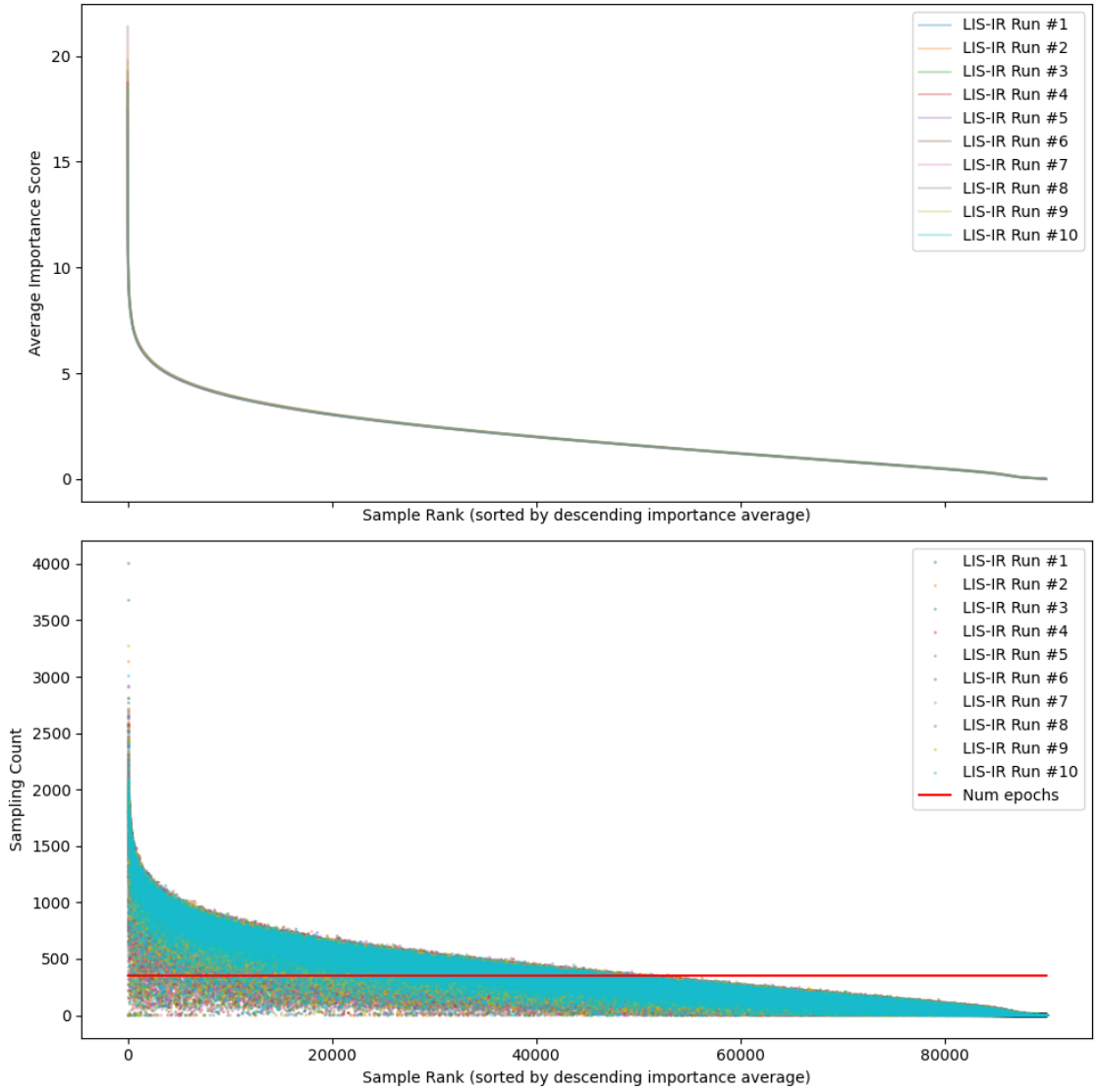

Figure B.3: Plot of each sample's training averaged importance score for each LIS-IR run (top plot), sorted in descending order, along with their corresponding sample frequency/count during training (bottom plot). For clarification; each x position corresponds to the sample in both subplots. The top plot is sorted by each sample's importance intra-run average. The bottom plot shows the corresponding sampling count i.e. the number of times it was sampled during training.

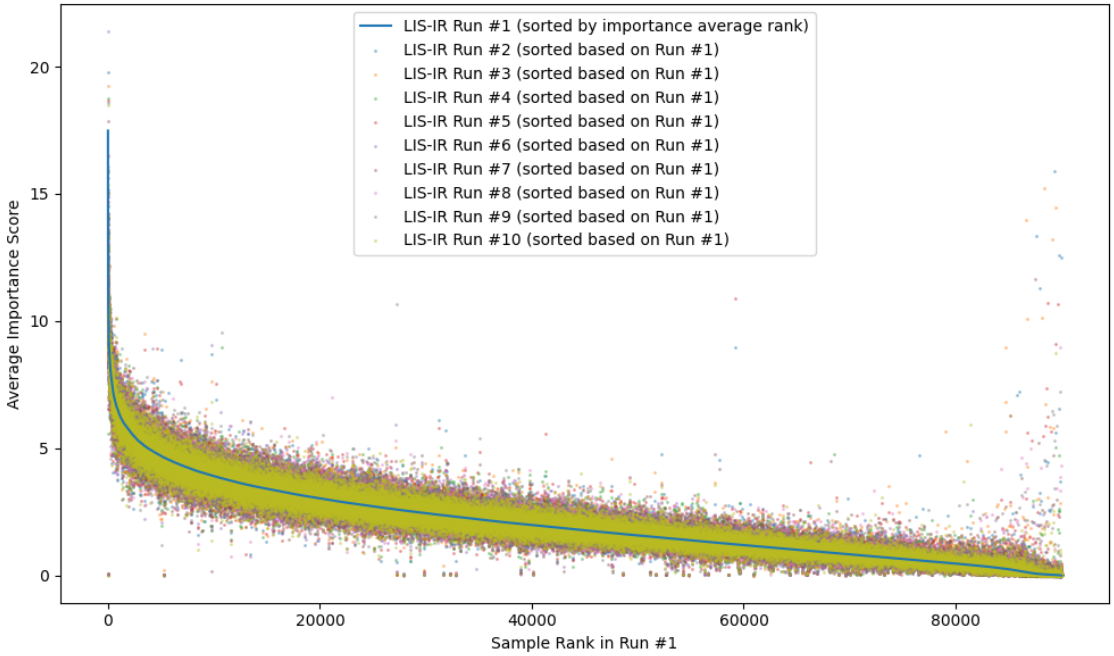

Figure B.4: Each sample's training averaged importance score across runs with (regression ignored) LIS sorted based on a single run.

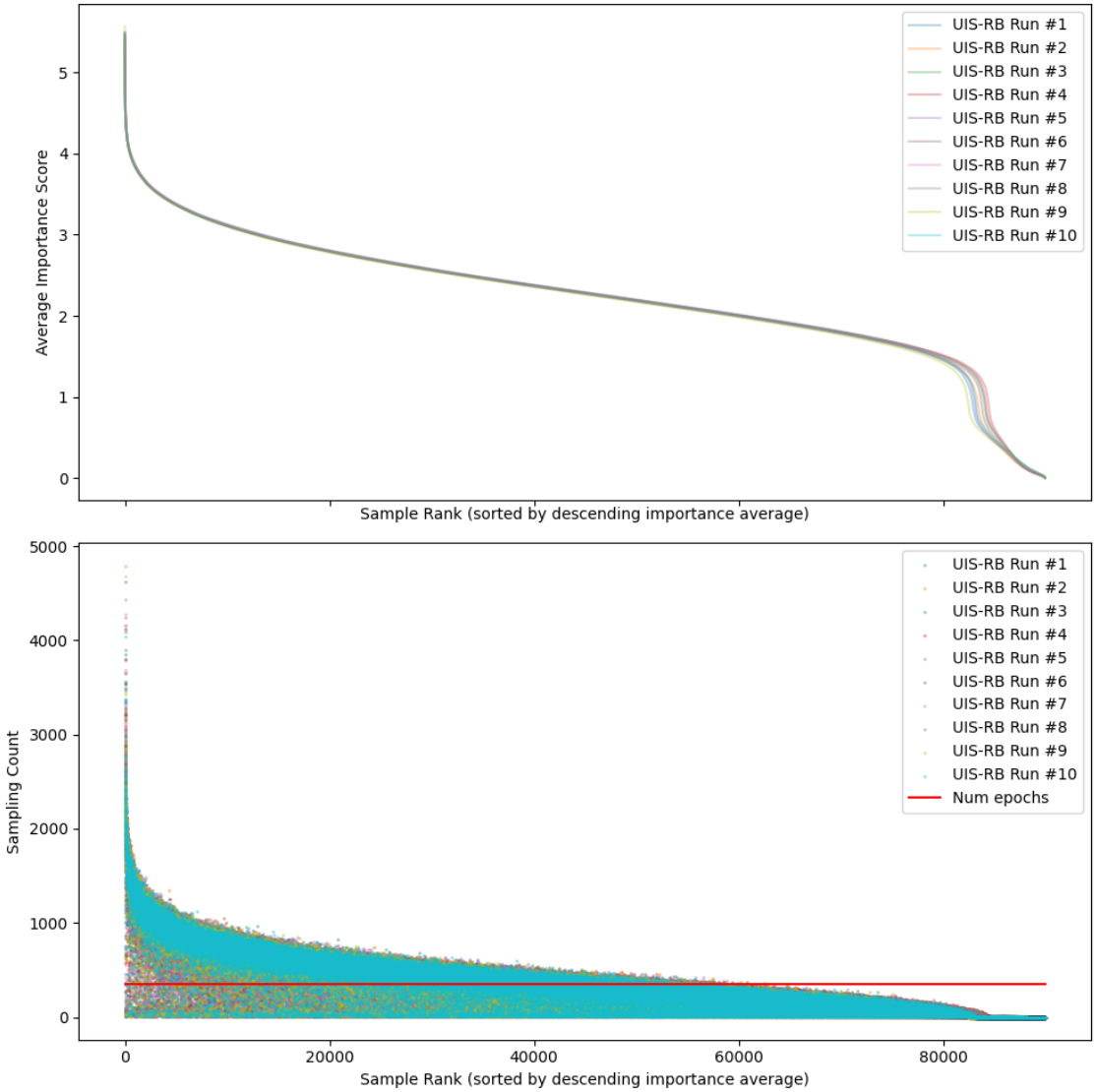

Figure B.5: Plot of each sample's training averaged importance score for each (rejection-based) UIS run (top plot), sorted in descending order, along with their corresponding sample frequency/count during training (bottom plot). For clarification; each x position corresponds to the sample in both subplots. The top plot is sorted by each sample's importance intra-run average. The bottom plot shows the corresponding sampling count i.e. the number of times it was sampled during training.

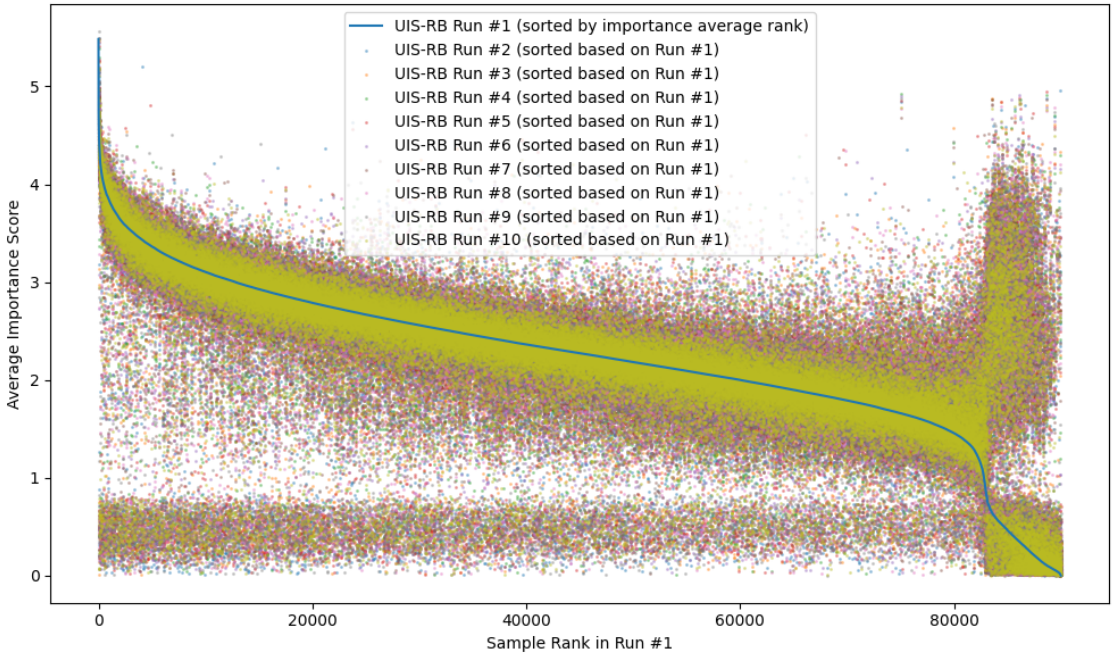

Figure B.6: Each sample's training averaged importance score across runs with (rejection-based) UIS sorted based on a single run.

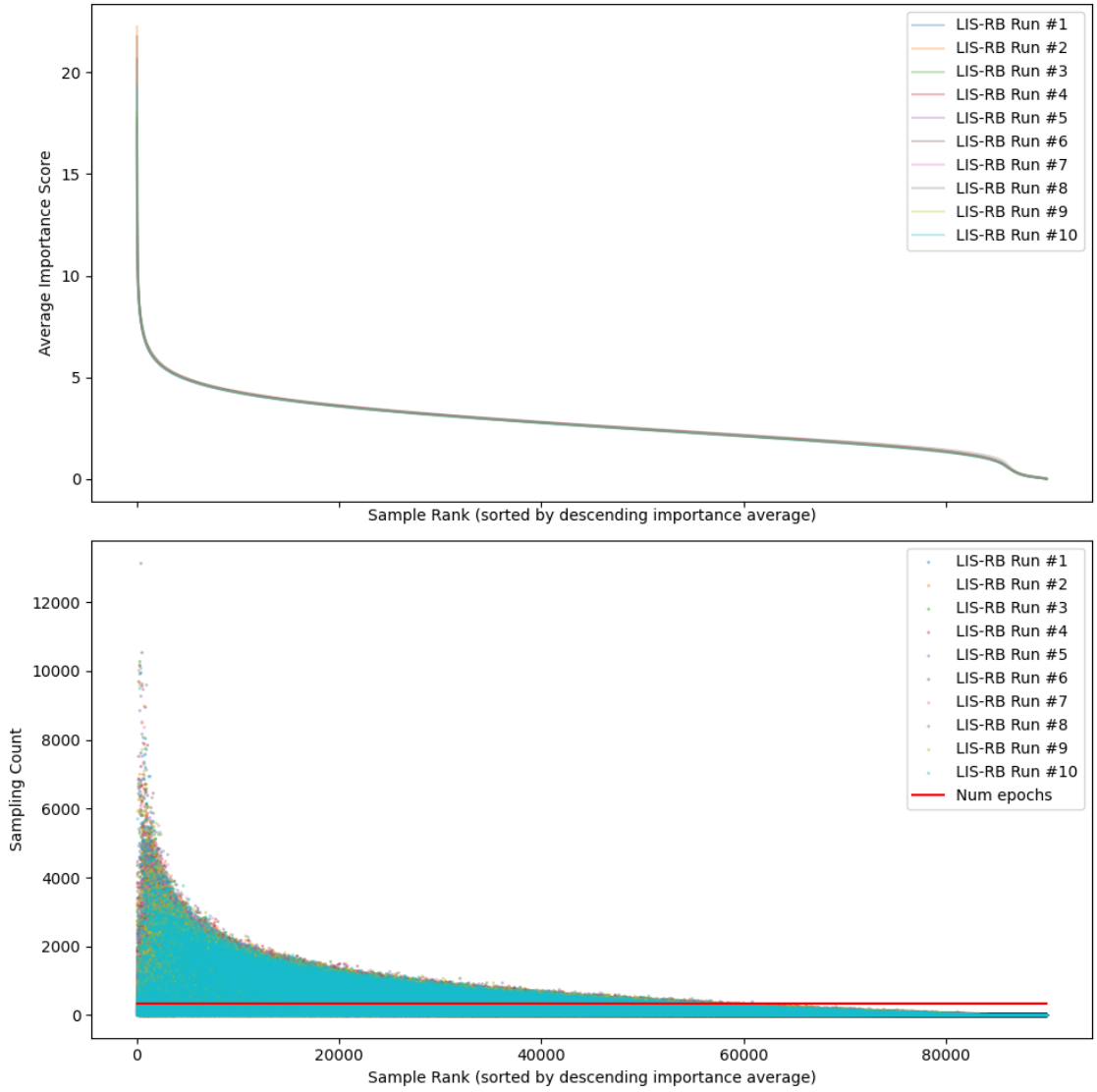

Figure B.7: Plot of each sample's training averaged importance score for each LIS-RB run (top plot), sorted in descending order, along with their corresponding sample frequency/count during training (bottom plot). For clarification; each x position corresponds to the sample in both subplots. The top plot is sorted by each sample's importance intra-run average. The bottom plot shows the corresponding sampling count i.e. the number of times it was sampled during training.

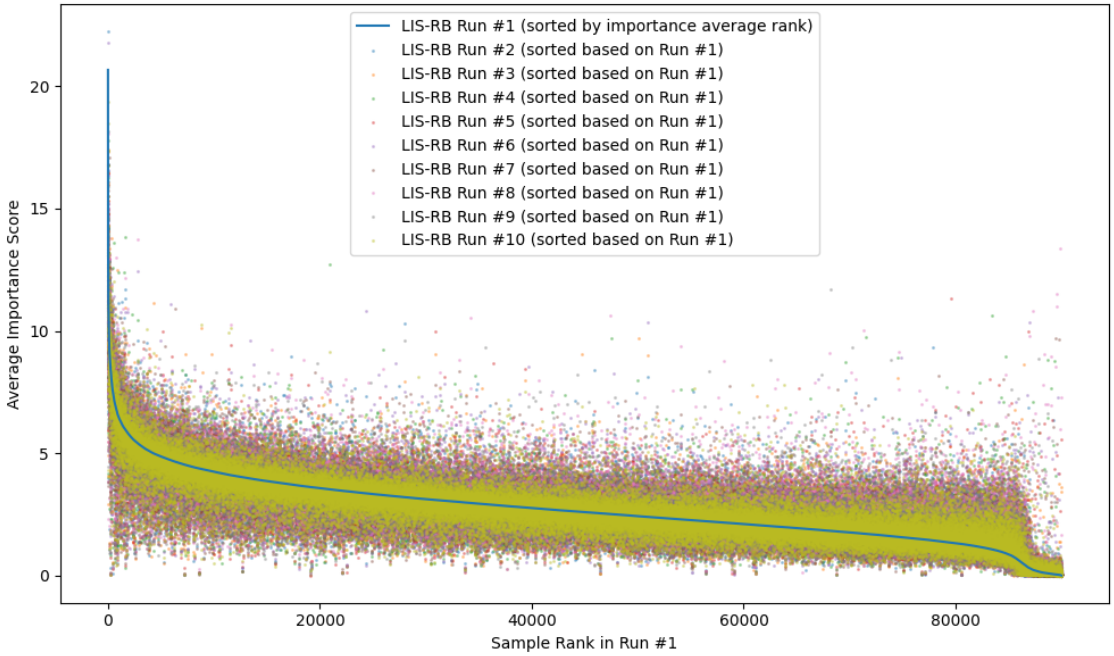

Figure B.8: Each sample's training averaged importance score across runs with LIS-RB sorted based on a single run.

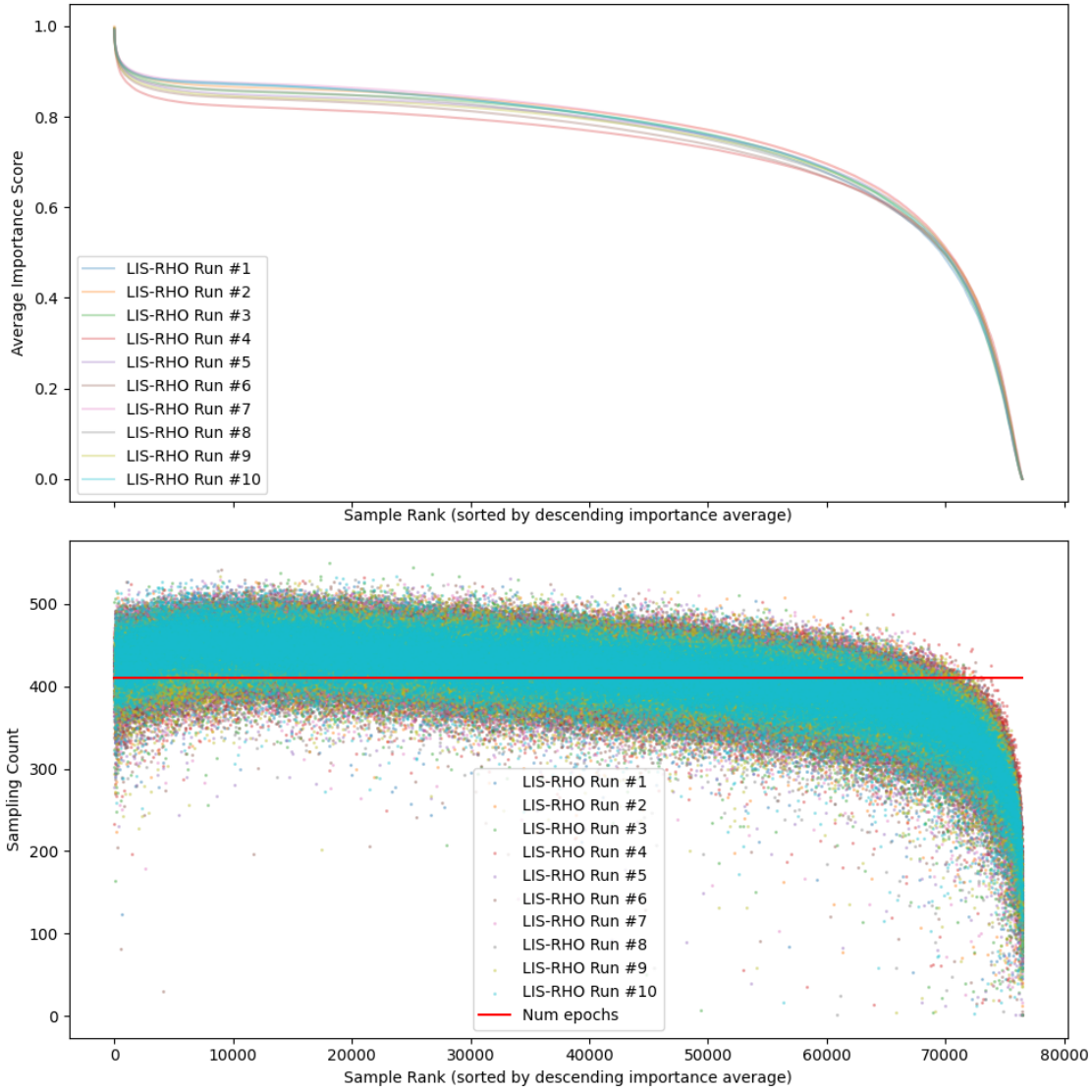

Figure B.9: Plot of each sample's training averaged importance score for each (RHO) LIS run (top plot), sorted in descending order, along with their corresponding sample frequency/count during training (bottom plot). For clarification; each x position corresponds to the sample in both subplots. The top plot is sorted by each sample's importance intra-run average. The bottom plot shows the corresponding sampling count i.e. the number of times it was sampled during training.

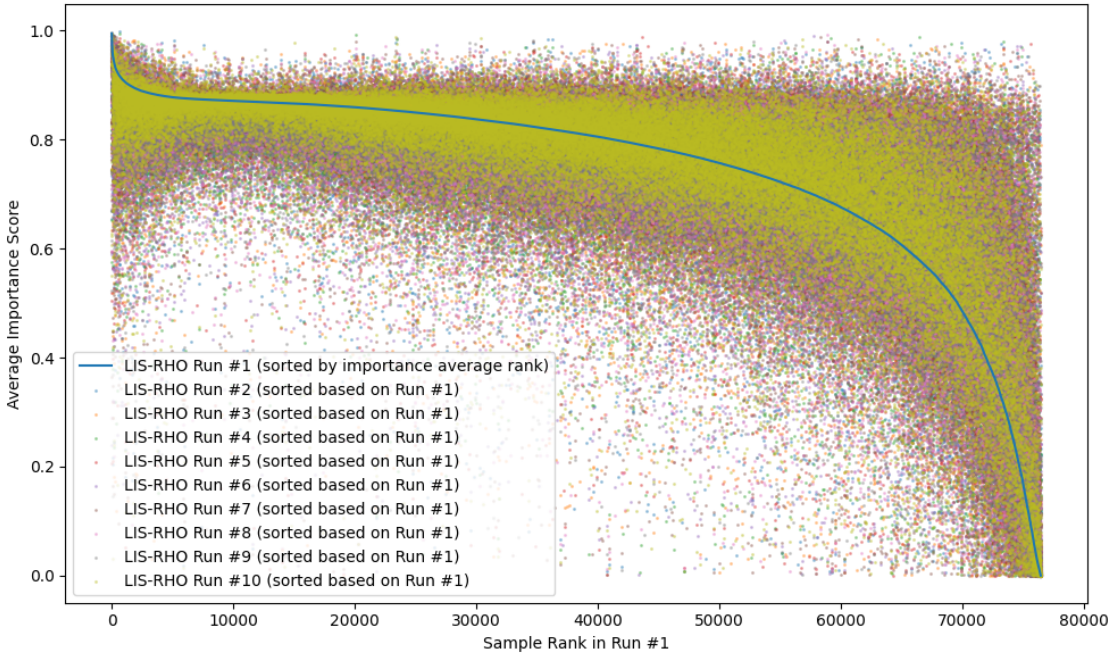

Figure B.10: Each sample's training averaged importance score across runs with (RHO) LIS sorted based on a single run.

C

Appendix 3

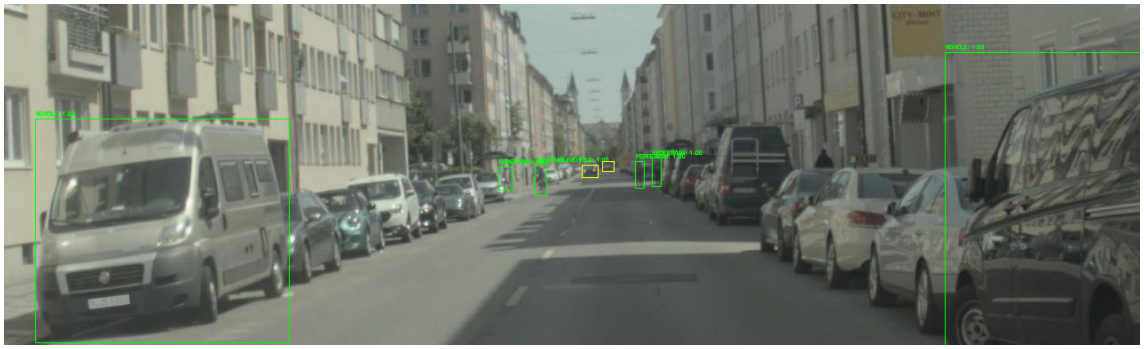

Figure C.1: An example of a mislabeled sample. Multiple vehicles have been left without a corresponding object annotation.

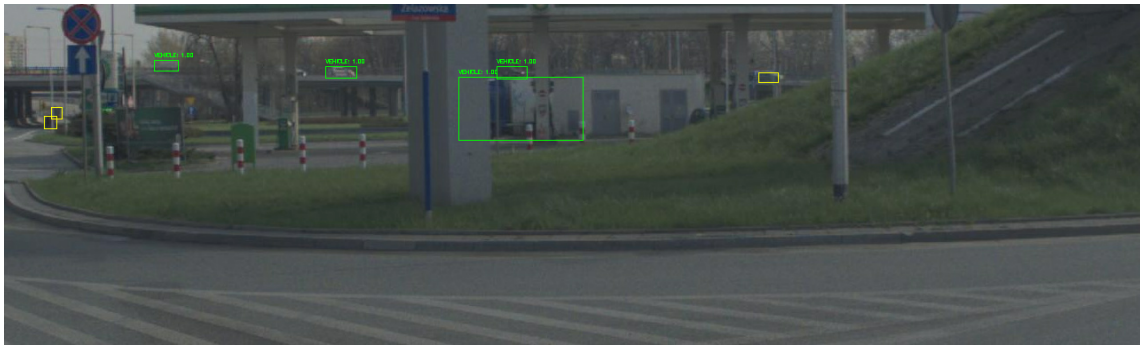

Figure C.2: An example of a mislabeled or overly ambiguous sample. It is unclear if the big object in the middle is a correctly labeled object that is overly ambiguous or if it is mislabeled.

Figure C.3: An example of an ambiguous sample. One object (identified by the car lights) is labeled correctly while some labeled objects consist of pixels that are too dark to decipher anything from, making them overly ambiguous.