Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Navjot Singh

# Deep Active Learning for Autonomous Perception

Master's thesis in Computer Science
Supervisor: Frank Lindseth, Håkon Hukkelås
June 2020

NTNU
Kunnskap for en bedre verden

NTNU | Norwegian University of Science and Technology

# Deep Active Learning for Autonomous Perception

Master's Thesis in Computer Science

**Navjot Singh**

Artificial Intelligence Group

Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering

Supervisor: Frank Lindseth
Co-Supervisor: Håkon Hukkelås

Spring 2020

# Abstract

Traditional supervised learning requires significant amounts of labeled training data to achieve satisfactory results. As autonomous perception systems collect continuous data, the labeling process becomes expensive and time-consuming. Active learning is a specialized semi-supervised learning strategy that allows a machine learning model to achieve high performance using less training data, thereby minimizing the cost of manual annotation.

In this thesis, we explore active learning in an autonomous domain, and propose a novel deep active learning framework for object detection and instance-based segmentation that comprises various active learning strategies. We review prominent active learning approaches, study their performances in the aforementioned computer vision tasks, and perform several experiments using state-of-the-art R-CNN-based models for datasets in the self-driving domain.

The goal of active learning is to let a machine learning model choose its own training data that helps it perform better. Using a query strategy, the model can measure the informativeness of an unlabeled sample based on its prediction uncertainty. The underlying theory is that by adding informative samples that are not represented in the current training data, the model will significantly improve. For a large set of training data and a well-trained model, these samples could represent highly unusual scenarios (i.e., a horse walking in the middle of a highway).

Our empirical experiments on a number of datasets reflect that active learning reduces the amount of training data required. We observe that early exploration with instance-rich training sets leads to good performance (i.e., active learning strategies using sum as aggregation technique), and that false positives can have a negative impact if not dealt with appropriately, especially for active learning strategies using average as aggregation technique. Furthermore, we perform a qualitative evaluation using autonomous driving data collected from Trondheim, illustrating that active learning can help in selecting more complex scenarios to annotate.

Encouraging findings indicate that active learning can be beneficial in a machine learning pipeline for autonomous perception to minimize the annotation job. Our efforts can be seen as an important first step for contributing in the active learning domain.

# Sammendrag

Tradisjonell veiledet læring krever betydelige mengder med annotert treningsdata for å oppnå tilfredsstillende resultater. Ettersom autonome persepsjonssystemer samler inn data kontinuerlig blir annoteringsprosessen meget kostbar og tidkrevende. Aktiv læring er en spesialisert semi-veiledet læringsstrategi som gjør det mulig for en maskinlæringsmodell å oppnå høy ytelse med mindre treningsdata, og minimerer dermed kostnadene for manuell annotering.

I denne oppgaven utforsker vi aktiv læring innenfor autonom persepsjon, og foreslår et nytt dypt aktivt lærings rammeverk for objektdeteksjon og instans-basert segmentering som omfatter ulike aktive lærings strategier. Vi gjennomgår fremtredende metoder innen aktiv læring, studerer deres ytelse i de sistnevnte datasyn oppgavene og utfører flere eksperimenter ved å bruke avanserte R-CNN-baserte modeller for datasett i det selvkjørende domenet.

Målet med aktiv læring er å la en maskinlæringsmodell velge sine egne treningsdata som hjelper den i å yte bedre. Ved hjelp av en spørrestrategi kan modellen måle informativiteten til et ikke-annotert datapunkt basert på dens prediksjonsusikkerhet. Den underliggende teorien er at ved å legge til informative datapunkter som ikke er representert i eksisterende treningsdata, vil modellen forbedre seg betydelig. For et stort sett med treningsdata og en godt trent modell kan disse datapunktene representere svært uvanlige scenarier (f.eks.. en hest som går midt på en motorvei).

Våre empiriske eksperimenter på en rekke datasett viser at aktiv læring reduserer mengden treningsdata som kreves. Vi observerer at tidlig utforskning med instanse-rike treningssett fører til god ytelse (dvs. aktiv læringsstrategier som bruker sum som aggregeringsteknikk), og at falske positive prediksjoner kan ha en negativ innvirkning hvis de ikke blir behandlet på riktig måte, spesielt på aktiv læringsstrategier som bruker gjennomsnitt som aggregeringsteknikk. Videre utfører vi en kvalitativ evaluering av autonome kjøredata samlet inn fra Trondheim, og illustrerer at aktiv læring kan være til hjelp med å velge mer komplekse scenarier for annotering.

Oppmuntrende funn indikerer at aktiv læring kan være fordelaktig i en maskinlærings prosess for autonom persepsjon for å minimere annoterings innsatsen. Vårt arbeid kan sees på som et viktig første skritt for å bidra til det aktive læringsdomenet.

# Preface

In this master thesis, I explore the specialized semi-supervised learning strategy Active Learning in an autonomous perception setting. This thesis is a part of an M.Sc. in Computer Science at Norwegian University of Science and Technology, and has been carried out within the NTNU Autonomous Perception Lab (NAPLab). I would like to express my sincere appreciation to my supervisors, Frank Lindseth and Håkon Hukkelås, for their continuous support and excellent guidance throughout this thesis, and for giving me this opportunity. Without them, I would not have been able to complete this thesis. I also wish to thank my mom, dad, and brother for all their love, encouragement, and moral support.

# Contents

# List of Figures

# List of Tables

# Acronyms

**AL** Active Learning. 2–5, 9–11, 14–18, 23–33, 35, 36, 41–43, 45, 49, 50, 52, 53, 55, 57, 58, 60, 66, 68, 69, 71, 72, 74, 75, 81–86, 91–104

**ALI** Active Learning Iteration. 2, 3, 17, 36–39, 41, 42, 94, 95, 99

**AV** Autonomous Vehicle. 1, 2

**BvSB** Best-versus-Second-Best. 13, 24, 25, 30, 38

**CIFAR** Canadian Institute For Advanced Research. vii, 5, 18, 19, 43, 49–51, 74

**CNN** Convolutional Neural Network. 1, 5, 23, 24, 33, 45

**COCO** Common Objects in Context. 6, 52, 69, 116

**CV** Computer Vision. 1, 3–5, 18, 23, 35, 38, 41, 98, 102

**DARPA** Defense Advanced Research Projects Agency. 1

**FCN** Fully Convolutional Network. 6, 7, 31, 70

**FPN** Feature Pyramid Network. 9

**IoU** Intersection over Union. 28, 29, 31, 32, 116

**MNIST** Modified National Institute of Standards and Technology. vii, 5, 23, 43, 45–49, 52, 58, 74

**QbC** Query by Committee. 12–14, 23, 27, 32, 33, 35, 44, 69, 70, 93

**ROI** Region Of Interest. 7–9

**RPN** Region Proposal Network. 6, 28, 29, 121

**SSD** Single Shot Multibox Detector. 6, 27, 28, 121

**SVM** Support Vector Machine. 23–27, 104

**YOLO** You Only Look Once. 6

# Chapter 1

# Introduction

In this chapter, the reader is given a short introduction to the background and motivation behind this thesis. In addition, our goals and contributions are presented, followed by an outline which describes the thesis structure.

## 1.1 Motivation

We have come a long way with the development of autonomous driving in the past years. One of the most significant milestones achieved was in 1987, when a vision-guided Mercedes-Benz van (VaMoRs) managed to reach a speed of 96 km/h in a non-traffic highway using dynamic vision techniques (Dickmanns, [14]). In 1989, the DARPA funded Autonomous Land Driven Vehicle (ALV) project, with the help of Computer Vision (CV) and other sensor technologies (e.g., LiDAR), managed to demonstrate the first off-road navigation in USA (Pomerleau, [60]). Today, we have several companies working with bleeding-edge state-of-the-art technologies for object detection, image segmentation, and advanced perception; in order to reach the common goal of making the first fully Autonomous Vehicle (AV).

Most of today's autonomous perception systems include various types of Convolutional Neural Networks (CNNs). CNNs are well suited for understanding, analyzing, and classifying visual imagery (O'Shea *et al.* [55]). Deep convolutional neural networks are being used for a large variety of CV tasks, but require a significant amount of labeled data to achieve satisfactory results. However, acquiring labeled data for all CV tasks is extremely expensive and time-consuming as it is often done by an expert human annotator. Making a label that classifies whether an image is of a car or pedestrian is easy. Drawing bounding boxes or classifying each pixel to a class in an image (i.e., segmenting), taken in an urban city environment, containing multiple objects, is much harder.

AVs generate large volumes of data, and annotating this can become a large bottleneck. Data is collected using multiple sensors, such as cameras,

LiDARs and RADARs, in just a couple of hours. Challenges arise when we have to deal with terabytes of data coming in every day, that has to be cleaned, processed, annotated, labeled, and stored in various ways for later use. With the help of a well-structured data pipeline, several techniques can be used to remove unnecessary, noisy, and corrupted data (Fridman *et al.* [19]), and to ensure that the data is consistent and synchronized during the process of sensor fusion. In a typical pipeline, data processing techniques are often fully automated. However, the annotation process is still done manually and has a high cost.

Active Learning (AL) is a specialized semi-supervised strategy that aims to minimize the annotation effort and maximize the usage of "highly useful" learning data (Settles, [68]). "Useful" data contain lots of new information that may be beneficial for the learner's understanding. AL gives the learner a chance to choose its own training data, where the goal is to achieve high performance using less data. This strategy has received much attention in the past recent years and fits well in scenarios where data is easy to obtain but expensive to label (Settles, [68]).

## 1.2 Aim

The goal of this thesis is to identify if AL can be used to achieve at least the same model performance using a smaller, carefully assembled dataset compared to using all of the data available. Furthermore, we will evaluate if AL can be used in a setting of autonomous driving by exploring and reviewing relevant AL approaches.

This thesis was requested by the NTNU Autonomous Perception Lab (NAPLab) to certify whether the field of AL is applicable for future development and worth focusing on in their current autonomous setting. NAPLab was started in 2018 and is a research group at the Norwegian University of Science and Technology (NTNU), which aims to research and develop state-of-the-art models for Autonomous Vehicles (AVs). Our contributions can be seen as an important starting point, since AL is an unexplored topic for NAPLab that needs to be investigated further.

To evaluate AL, we will implement a pool-based AL Framework to perform object detection and instance-based segmentation tasks. We aim to use this framework with Detectron2 (Wu *et al.* [82]) on the Apollo Synthetic dataset [74], and preferably on a real-life dataset (e.g., Waymo Open [80]). This will be done in an iterative fashion by implementing simpler frameworks using less complex models and datasets to verify and support subsequent implementations.

Various AL query strategies will be evaluated based on their performance and compared to a baseline method. We will be using the general procedure of AL as follows: A model is trained for a number of Active Learning Itera-

tions (ALIs) using each query strategy. During each ALI, the trained model queries a set of unlabeled samples. Each sample is given an informativeness score that is measured by a query strategy. This score tells us how certain or uncertain a model is about a sample. If a model is uncertain about a sample, it considers the sample as being highly informative. A set of highly informative samples is added to the existing training set, and the model is then re-trained in the following ALI.

## 1.3 Research Questions (RQs)

The overall goal is to implement an AL Framework that we can use to explore and apply AL, and to evaluate the performance of this strategy in various CV tasks using a number of self-driving datasets. To investigate this, we will do an evaluation by looking into the following research questions:

**RQ1:** Can we use AL to achieve better or similar performance with less labeled data as opposed to utilizing the entire dataset? If so, how much time and resources can be saved in collecting, annotating data, and training in the setting of autonomous perception?

**RQ2:** Does there exist an optimal query strategy that can be used for either object detection and/or segmentation in the setting of autonomous perception?

**RQ3:** Does AL perform differently depending on the CV task, or is there a similar performance pattern? Does it work well with instance-based segmentation?

## 1.4 Contributions

In this thesis, we propose a novel AL Framework that contains various AL strategies to be used in object detection and instance-based segmentation. The framework uses Detectron2's implementation of the state-of-the-art Faster R-CNN and Mask R-CNN models (Wu *et al.* [82]).

#### Other Contributions

- We present, evaluate, and implement various AL strategies to see if they are applicable for object detection, and instance-based segmentation related to autonomous driving.
- We implement dropout layers, early stopping hooks, and evaluation hooks in Detectron2, as Detectron2 is a relatively new library that does not include this functionality out of the box.

- We implement an algorithm for data diversification that uses the similarity metric LPIPS [86].
- We present an AL Pipeline for our AL Framework and explain how it can be used in an autonomous setting as future work.
- We make self-driving datasets, such as Apollo Synthetic, Waymo Open, and NAPLab's raw data compatible with Detectron2 so they can be used for future work.
- As for the near future, we have ambitions to publish a paper on the work presented in this thesis.

## 1.5 Outline

This thesis is organized as follows:

**Chapter 2: Background Theory** looks into relevant theory related to CV tasks, state-of-the-art models, AL, and datasets.

**Chapter 3: Related Work** gives a brief overview of some related work in the field of AL; AL in image classification, object detection, and segmentation.

**Chapter 4: Methodology** presents our AL Framework, the various AL strategies being used, and the structure of our experiments.

**Chapter 5: Experiments and Results** presents the results of the various experiments conducted using different AL strategies in our iterative development.

**Chapter 6: Discussion** discusses our major findings and the RQs.

**Chapter 7: Conclusion and Future Work** concludes and reflects this thesis, proposes an AL pipeline and presents new ideas for future development.

# Chapter 2

# Background Theory

This chapter gives the reader a brief introduction to the relevant theory for this thesis. It is assumed that the reader has the necessary prior knowledge about Deep Neural Networks and Convolutional Neural Networks. We will start by presenting the evolution of common Computer Vision (CV) tasks for autonomous vehicles, including some of their current state-of-the-art models. Furthermore, we will explain the specialized semi-supervised strategy Active Learning (AL), how it is implemented in machine learning and how it can benefit from techniques such as early stopping, data augmentation, and data diversification. Finally, we will list relevant datasets that are used in the CV field and this thesis.

## 2.1 Computer Vision Tasks

We will specifically focus on image classification, object detection, and instance-based segmentation, due to their vast usage in autonomous perception.

### 2.1.1 Image Classification

Image classification is the task of classifying an object in an image to a specific class. For instance, a CNN can be built and trained to classify whether an image is of a vehicle or a pedestrian. It is possible to classify multiple objects in an image. However, the output will only consist of classifications, and will say nothing about the localization of each object.

There exists several datasets that are used to train classification models, one of the most popular being CIFAR-10/100 [43], ImageNet [13], and MNIST [44]. State-of-the-art classification models (e.g., ResNet [30]) excels at this task, often outperforming humans.

### 2.1.2 Object Detection

Object detection is a combination of localization and classification, where the goal is to detect objects by classifying them as a class and localizing their positions. The output is typically a set of bounding boxes with corresponding classifications and probabilities of how certain the predictions are. Figure 2.2a illustrates a typical object detection output.

The task of object detection has gained traction in recent years, with large-scale labeled datasets being a key driving force. PASCAL VOC [17] was a key factor for early deep learning models, and recently, challenging datasets, such as COCO [48] and ImageNet [13], has further driven the field.

Object detection models are often categorized into two models; real-time detection (e.g., Single Shot Multibox Detector (SSD) [49], You Only Look Once (YOLO) [63]), and more expensive models. In this thesis, we will not limit ourselves by computation time; therefore, we will focus on the R-CNN branch of models.

#### Faster R-CNN

Faster R-CNN (Ren *et al.* [64]) is an object detection network based on R-CNN [26] and Fast R-CNN [25]. In short, the network takes an image as input and outputs a bounding box and class prediction for each detected object in that image.

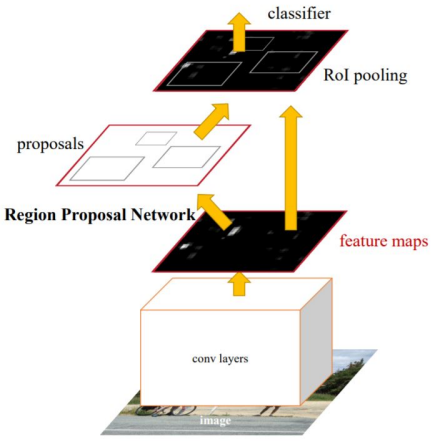



**Figure 2.1:** An illustration of the Faster R-CNN network. Figure source: Ren *et al.* [64]

An image is given as input to a Fully Convolutional Network (FCN) that outputs a feature map of the image. This feature map is used by a Region Proposal Network (RPN) to find areas that might contain objects, and this is done in a sliding-window fashion. For each window, $k$ number of anchor boxes are generated. For each anchor box, two outputs are given; an

objectness score in the form of a binary class label that tells whether there is an object in this anchor or not, and an initial proposal of the bounding box for the object. Multiple Region Of Interests (ROIs) are proposed using these outputs.

ROI proposals are used by the ROI pooling layer along with the feature maps from the FCN. ROI pooling crops each proposal to make it contain a single object. These proposals are then fed into a classifier that finally classifies the object and refines the initial bounding box proposal.

### 2.1.3 Image Segmentation

The task of image segmentation is to analyze an image and split it into multiple segments that represent different object classes. This is done in a pixel-wise manner, where we label each pixel as a specific class. It can, therefore, be seen as a simplification of the image, which is easier to understand and faster to analyze for later processing. This technique can help us detect and easily distinguish different types of objects in an image.

When an image is split into segments, each segment can be classified as either a thing class or a stuff class [10]. Following the definitions by Caesar *et al.* [10]:

**Thing Classes** are defined as having characteristic shapes, characteristic sizes, identifiable parts, and being countable.

**Stuff Classes** are defined as being amorphous, variable in size, having non-identifiable parts, non-countable, and highly textured.

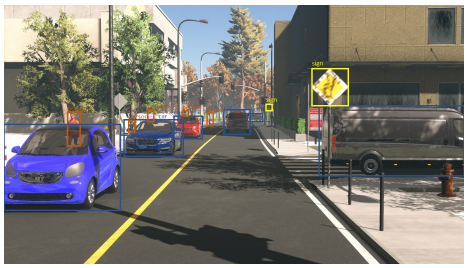
The three methods semantic segmentation, instance-based segmentation, and panoptic segmentation, use these classes differently.

In **Semantic Segmentation**, the task is to classify each pixel as a specific class. The image is split into different segments at a pixel level based on their class, and these segments are grouped. For instance, an image containing five overlapping vehicles will be classified as a single thing class segment, as illustrated in Figure 2.2b.

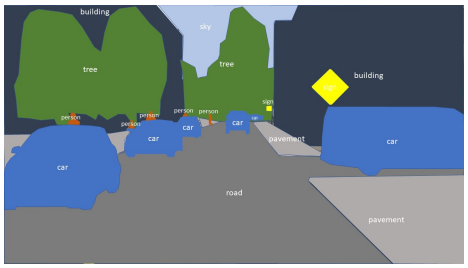
**Instance-Based Segmentation** is a combination of semantic segmentation and object detection. Here, we care about each instance of a class object. Let us say we have an image containing three overlapping vehicles. As mentioned earlier, a semantic segmentation will classify them as a single stuff class. Instance-based segmentation will make sure that it outputs and differentiates each vehicle as a single segment, making it a thing class, as seen in Figure 2.2c. Instance-based segmentation does not look into stuff classes. Networks such as Mask R-CNN [31] and YOLOACT [7] can perform this task.

**Panoptic Segmentation** unifies semantic segmentation and instance-based segmentation [42]. Each pixel in an image is classified as either a

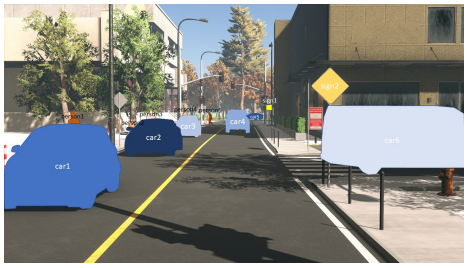
thing class or a stuff class; see Figure 2.2d. Since stuff classes are difficult to quantify, they are not split into instances, unlike thing classes.



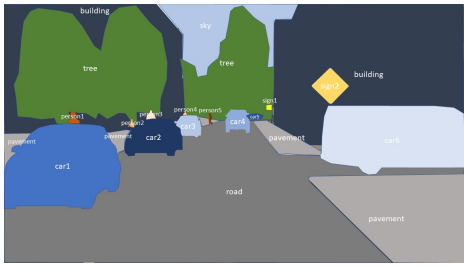
**(a)** Object Detection




**(b)** Semantic Segmentation




**(c)** Instance-Based Segmentation




**(d)** Panoptic Segmentation

**Figure 2.2:** Example outputs from object detection and segmentation tasks.
[2.2a] Each detected object has a bounding box, including its predicted class and probability.
[2.2b] Each pixel is assigned to a segment, including its predicted class and probability.
[2.2c] Each detected instance of an object class has its unique segment, including its predicted class and probability.
[2.2d] Each pixel is assigned to a segment having its own instance, including its predicted class and probability.

### Mask R-CNN

Mask R-CNN (He *et al.* [31]) is a framework that can perform object instance-based segmentation, and is an extension of Faster R-CNN [64]. This is done by adding a new branch that can predict segmentation masks on each Region Of Interest, where the predicted object mask is generated in parallel with the bounding box generation.

Mask R-CNN replaces RoI-pooling with RoI-align, to combat the issue of misaligned region proposals. This misalignment was not acceptable since the segmentation layers required a pixel-level specificity. They resolved this issue by transforming the ROI pooling into ROI align that avoids rounding

on these regions. The backbone was improved by using a Feature Pyramid Network (FPN) [47], which increased the speed and accuracy.

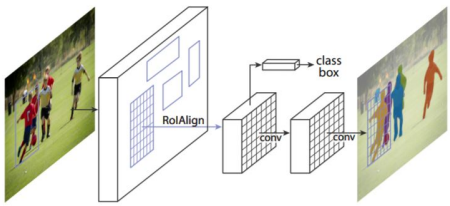



**Figure 2.3:** An illustration of the Mask R-CNN architecture. Figure source: He *et al.* [31]

In parallel to the original Faster R-CNN classifier and bounding box regressor, soft masks are generated for each ROI. The soft masks contain float numbers, which makes them more detailed. They are later converted into binary masks that give us the final segmentations.

#### Detectron2

Detectron2 is FAIR's (Facebook Artificial Intelligence Research) latest open-source research platform that contains state-of-the-art algorithms that can perform object detection and segmentation (Wu *et al.* [82]). This platform includes models such as Faster R-CNN, Mask R-CNN, Cascade R-CNN, Panoptic FPN, TensorMask, and DensePose. Specifically, we will focus on the platform's implementation of Faster R-CNN and Mask R-CNN.

## 2.2 Active Learning

Active Learning (AL) is a semi-supervised learning strategy that aims to train a machine learning model to achieve high performance using less but "highly informative" training data. From a model's perspective, informative data is data that the model is uncertain about and struggles to predict; AL introduces various techniques that can measure a model's prediction uncertainty. In AL, a machine learning model is given the opportunity to select its own training data that helps it perform better (Settles, [68]). By letting the model select a set of highly informative datapoints, its understanding of the current environment might improve (Settles, [68]). More importantly, by using less training data, AL reduces the cost of manual annotation.

In this section, we will present the different parts of an AL Framework, following Settles [68], which includes multiple Query Scenarios, Query Strategy Frameworks, Query Strategies, and Sample Selection Strategies. In addition, we will present some weaknesses in AL and how techniques such as early stopping, data augmentation, and data diversification tackles these.

Throughout this thesis, a machine learning model that is being trained using AL is defined as a "learner."

### 2.2.1 Query Scenarios

A query scenario can be referred to as a sampling technique. It presents a specific way for a learner to ask an annotator for labels on highly informative samples (Settles, [68]). Every time a highly informative sample is queried, it is added to the training set, which is used to train the learner. There exist mainly three sampling techniques:

- **Membership Query Synthesis:** The learner can either ask for labels of an unlabeled sample, or generate a new sample from the underlying natural distribution of the data, which has to be then labeled by an annotator. For instance, if the dataset contains images of vehicles, the learner has to generate an image that resembles a vehicle. An underlying natural distribution of this data has to be guaranteed to ensure that the learner can generate an understandable sample, which helps the annotator to annotate the sample successfully.

- **Stream-Based Selective Sampling:** The learner expects an unlabeled set of samples that it can inspect. Each sample is selected one at a time from the unlabeled set, and the learner decides whether or not to ask for the label of that sample. This gives the learner a choice to reject samples with already known labels or to query labels of samples being highly informative. Each sample is queried or discarded based on a threshold, and finding a good threshold is crucial for this scenario to work. This technique can become computationally heavy since the learner has to be trained each time a single sample gets selected.

- **Pool-Based Sampling:** The learner expects an unlabeled set of samples that it can inspect. All samples are selected from the unlabeled set to measure their informativeness, but only a small set of highly informative samples gets queried to be labeled by an annotator. Finding an adequate size of samples to be queried can become a challenging task. A drawback with this technique is that a learner has to inspect the whole unlabeled set after each training to select a set of highly informative samples. However, there exist techniques that can cope with this issue (Ertekin *et al.* [16]), which will be covered in Chapter 3. An illustration of this sampling technique is shown in Figure 2.4.

Since we will be working with complex deep neural networks, a pool-based sampler is highly favored compared to a stream-based. Training a network repeatedly by adding only a single sample at a time is not feasible, as this can become computationally heavy and time-consuming. Therefore,

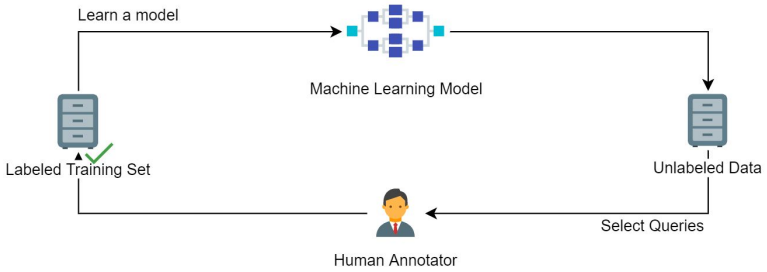



**Figure 2.4:** A representation of a pool-based AL cycle. The learner selects a small set of highly informative samples from the unlabeled set for the annotator to label. The labeled samples are then added to a training set, which the learner is trained with. This figure is based on Settles [68].

using pool-based sampling, batches of highly informative samples are added to the training set instead.

### 2.2.2 Query Strategy Frameworks

A query strategy framework is used to acquire some sort of informativeness measurement from unlabeled samples (Settles, [68]). To be clear, a query scenario only defines the way of querying labels of samples by either generating them, selecting one at a time, or as a batch (e.g., membership query synthesis, stream-based, or pool-based). Samples are queried based on their informativeness that is measured by this framework. There exist several query strategy frameworks:

#### Uncertainty Sampling

Uncertainty sampling is the most used and simplest framework (Settles, [68]). Here, a single learner is used to query labels of samples which it is uncertain of how to label. Since its uncertainty comes from the prediction uncertainty, a probabilistic learning model is required by the framework.

For instance, let us say we have a set of unlabeled samples. Using pool-based sampling, the learner runs inference over the entire unlabeled set of samples. The prediction results are then used by a query strategy to measure the informativeness score of each sample. These scores give the learner a clue on which set of highly informative samples to query.

#### Query-by-Committee (QbC)

A committee of learners is used to query labels of samples which they disagree the most (Settles, [68]). For instance, let us say we have a set of unlabeled samples. Using pool-based sampling, a committee of learners is

used to run inference over the entire unlabeled set of samples. Their prediction results are compared. If their results vary, it is a sign of disagreement, and the samples of which they disagree the most are considered to be highly informative.

Training a committee of learners can be computationally heavy and time-consuming. When using deep learning models, this framework can be simplified to virtually resemble a committee with the help of dropout layers (Gal *et al.* [20]). By using these layers on a single model, we can run inference multiple times over each sample and measure the variations in predictions to measure uncertainty (Morrison *et al.* [54]). Having a different output during each inference shows high uncertainty.

#### Other Frameworks

Other existing frameworks (Settles, [68]) are listed below with increasing complexity:

- **Expected Model Change:** Selecting samples that would impact and influence the current learner greatly if we knew their labels. An example framework is "expected gradient length," where samples that would change the gradient the most are queried.
- **Expected Error Reduction:** Selecting samples that would give a minimal expected future error, or that would reduce the total number of incorrect predictions in the future.
- **Variance Reduction:** Selecting samples that would give a minimal output variance.
- **Density-Weighted Methods:** Selecting samples that the learner is uncertain about, but make sure that they are representative of the underlying distribution of the data.

Most of these frameworks are computationally heavy and complex. Therefore, they are not looked very much into in detail. For object detection (i.e., Faster R-CNN), we will be mainly focusing on using a traditional uncertainty sampling framework, as it is easy to use, has a low compute time which balances out the computationally heavy R-CNN model, and fits well with a setting for object detection, which uses probability distribution over classes to measure uncertainty. In addition, for both object detection and instance-based segmentation (i.e., Mask R-CNN), we will be using a virtual QbC framework for measuring uncertainty.

### 2.2.3 Query Strategies

A Query Strategy is used by a Query Strategy Framework, such as Uncertainty Sampling, to measure the informativeness of a sample. Keep in mind

that most of these strategies are mainly related to probabilistic learning models, and require having a probability distribution over all classes (i.e., the output in the form of a softmax vector), which is what we will be using. Several query strategies exist, but the most common are the ones listed below:

- **Random Sampling (R):** The learner selects samples that are randomly selected from the unlabeled set.
- **Least Confidence Sampling (LC):** The learner queries samples which it is least confident about.

$$x_{LC} = \underset{x}{\operatorname{argmax}}(1 - P(\hat{y}|x)) \tag{2.1}$$

  $x$ is a sample and $\hat{y}$ is the highest predicted class probability of that sample. Samples with the least confidence on its most probable label are selected.

- **Margin Sampling (M):** The learner selects samples based on their class probability margins.

$$x_M = \underset{x}{\operatorname{argmin}}\, P(\hat{y_1}|x) - P(\hat{y_2}|x) \tag{2.2}$$

  This strategy looks at the highest probable class label and the second-highest probable class label (i.e., $\hat{y_1}$ and $\hat{y_2}$, respectively) for each sample. The margin between these two labels is then calculated, where smaller margins indicate high informativeness. Joshi *et al.* [36] propose this margin sampling measure, which they refer to as the Best-versus-Second-Best (BvSB) approach.

- **Entropy Sampling (E):** The learner selects samples that give the highest entropy.

$$x_E = \underset{x}{\operatorname{argmax}} \sum_i P(y_i|x) \log P(y_i|x) \tag{2.3}$$

  $i$ ranges over all class probabilities of the sample $x$. Higher entropy score shows higher informativeness between class labels.

The listed Query Strategies can be used with a QbC Framework, assuming that a probability distribution over all classes can be obtained. This can be done by simply comparing the informativeness of each sample from each learner and measure their variation; the higher their variation, the higher their disagreement.

For instance, let us say we have a committee of learners that are used for detecting a single object in an image. If each learner outputs a softmax vector having different predicted class probabilities, it is a sign of disagreement, and the image is considered to be highly informative.

Other query strategies for QbC, such as measuring segmentation uncertainty using pixel-probability and dropout, are explained in detail under Section 3.3.

#### Strategy Aggregation

The query strategies mentioned earlier are highly generalized and work well for image classification, where each image is fed into a neural network that outputs a softmax vector containing the required class probabilities. Since our focus is on object detection and segmentation, an approach is needed to measure the informativeness of the entire image based on the informativeness of every detection in that image. This can be achieved by using aggregation techniques. Roy *et al.* [66] and Brust *et al.* [9] apply similar techniques in their approaches. For instance, in object detection, an image can be scored by aggregating the measured informativeness of all detections. In segmentation, an image can be scored by aggregating the measured informativeness of all segments. Since each detection in an image is given an informativeness score, the scores can be aggregated in different ways:

- **Sum (SUM):** Taking the sum of all detection scores in an image and setting it as the image score.

$$Image_{QS_{SUM}} = \sum_{i=1}^{n} x_{QS_i} \tag{2.4}$$

  Here, $QS$ is the chosen query strategy, $n$ is the total number of scored detections in an image and $x_{QS_i}$ is the score of the current detection $i$.

- **Maximum (MAX):** Taking the maximum detection score in an image and setting it as the image score.

$$Image_{QS_{MAX}} = \max_{i} x_{QS_i} \tag{2.5}$$

- **Average (AVG):** Taking the average of all detection scores in an image and setting it as the image score.

$$Image_{QS_{AVG}} = (\sum_{i=1}^{n} x_{QS_i})/n \tag{2.6}$$

### 2.2.4 Sample Selection

During the AL cycle (Figure 2.4), the learner has to select which samples to query based on their informativeness. Samples can be selected in several ways, but the most common way is to select the top $k$ samples having the

highest informativeness. Other selection methods may select samples from the whole informativeness spectrum.

Roy *et al.* [66] uses a method that they call "n bins formulation." Scored samples are split into batches with size $m$. In each batch, the score space is divided into $n$ bins. $m/n$ samples with the highest informativeness are selected from the top $n-1$ bins, and $m/n$ from the last bin having the lowest informativeness. They use this method to cope with the exploration vs. exploitation dilemma by getting the best of both, which is explained in more detail in Section 2.2.6.

### 2.2.5 Known Weaknesses in AL

In most scenarios, when using AL, a model has to be trained on a never-before-seen dataset. Due to the iterative nature of AL, the model is trained with an increasing training set. Here, the distribution of classes can have a huge impact on the model's performance. Some weaknesses are listed below:

- **Not Understanding the Underlying Distribution:** If the training set does not contain enough information about the underlying distribution of the whole dataset, a model can struggle to understand the data. This can result in poor performance and may delay the learning process.

- **Class Bias:** Some sample methods tend to collect more samples from the majority classes than the minority classes. This can over-time lead to an unbalanced training set. Thereby making the model more biased. Ertekin *et al.* [16] explores this issue in the context of image classification, and their results show that this does indeed impact the prediction performance.

- **Complex and Noisy Samples:** Several studies [35, 36] work with binary classification. An assumption can be made that the data is separable for binary classification by finding a threshold. The goal is to collect highly informative samples near this threshold that represent the underlying distribution of the data. However, some of these samples can be highly complex. They can cause the training to converge wrongly, and the model's confidence in the underlying distribution can be misplaced.

Trade-offs need to be considered when choosing the right combination of AL strategies (e.g., query scenarios, query frameworks, query strategies, aggregation strategies), as this can have a significant impact on the AL Framework's complexity and cost.

### 2.2.6 Additional Techniques that can be used with AL

Various techniques that can be beneficial to be used with AL have been proposed:

#### Exploration and Exploitation

The exploration vs. exploitation trade-off can be a difficult problem to cope with. The optimal goal is to acquire more knowledge and gain maximum reward at the same time. The learner has to either *explore* new unknown problems to gain greater rewards, or to *exploit* the current situation and make the best out of it.

In AL, this is a known dilemma, since the learner starts with little to no knowledge about the data (Bondu *et al.* [8]). For instance, doing exploration initially, will be crucial to explore the feature space and classes in the dataset. When the network has gained enough knowledge, it can start exploiting. During exploitation, the network can fine-tune itself and make itself even more capable of classifying correctly, based on what it already knows.

Bondu *et al.* [8] investigates the exploration vs. exploitation dilemma, and illustrate this issue with two extreme cases; An active learner who only exploits can become very specialized in one area, but can fail to understand the whole data space. An active learner who only explores can miss the focus on important data, which might be beneficial for the model's performance. They present three common approaches to tackle this problem:

- **Use of multiple strategies** by, for instance, switching between a random sampler, which is good at exploring, and other uncertainty sampling methods that can be used for exploiting.
- **Pre-clustering** data by only choosing cluster centroids as highly informative candidates that are a part of the same class, if possible.
- **Use of similarity measure** by ensuring that the data is diverse.

#### Data Diversity

Data diversity is important and can have a positive impact on a model's performance (Gong *et al.* [27]). In AL, the goal is to achieve high accuracy using less training data. This data has to be diverse since the learner prefers samples that are more useful and less redundant. It prevents the model from becoming biased towards a specific class or a set of classes. Having a carefully selected, highly informative, and diverse initial training set would be a perfect scenario as this could give us a good performing model early in the AL process.

### Image Similarity

There exist several open self-driving datasets, and most of them contain a set of video sequences having thousands of frames. There is always a chance that these sequences might contain duplicates or highly similar frames, which can become a problem. The usage of a similarity measure was proposed by Bondu *et al.* [8] to tackle the exploration vs. exploitation dilemma as well as ensuring data diversity.

To give an example, let us say we are training a model using AL. During each ALI, the model has to query useful samples to be labeled. Assume that this set of samples contains a set of frames from a video sequence. If the learner finds an object in a frame that it is highly uncertain of, there is a high chance that this object might appear in the surrounding frames. The learner might ask the oracle to label a set of highly similar frames due to its uncertainty. This does not lead to data diversity. The following metrics can be used to prevent the overuse of similar images:

- Structural Similarity Index (SSIM) (Wang *et al.* [88]). It extracts structural information from an image and can be used to compare images.
- Learned Perceptual Image Patch Similarity (LPIPS). Zhang *et al.* [86] uses deep neural network activations as a perceptual similarity metric.

### Transfer Learning

Transfer learning gives us the ability to use a previously trained model as a starting point [57]. A model that is developed to perform a specific task in a domain of interest can be used in another similar domain. Kale *et al.* [37] presents how transfer learning can be used to accelerate AL, by initializing an active learner using transfer learning, and training it with labeled data from similar tasks to cope with the cold start problem.

### Early Stopping

Early stopping is a simple tool that prevents a model from overfitting [61]. Ertekin *et al.* [16] improve their results with early stopping in a class imbalance problem using AL.

### Dropout and Monte Carlo Dropout

Dropout is a tool that is applied during training to prevent a model from overfitting and to increase its robustness [72].

Gal *et al.* [20] proposes Monte Carlo dropout (MC dropout) to measure the uncertainty of a model. MC dropout applies dropout during both training and inference. By running inference over samples multiple times using

MC Dropout, they are able to measure the difference in predictions. High difference is a sign of high model uncertainty.

#### Data Augmentation

Data augmentation can be used to expand the dataset artificially by augmenting images to make new samples [70]. This technique has been widely used in various CV tasks and when working with limited data resources, which is a common scenario in AL. Augmenting data can, in some scenarios, increase the performance of a deep learning model and prevent overfitting [59].

## 2.3 Datasets

Datasets play a fundamental role in machine learning. Having a rich and balanced set containing lots of high-quality labeled data is desired. This section gives a quick introduction to the different datasets that are used with various CV tasks and with AL. The reasons behind our dataset choices are explained in more detail under each experiment in Chapter 5.

### 2.3.1 MNIST

MNIST is an accessible "hello world" database built for image classification tasks. It contains a total of 70 000 grayscale images of handwritten digits ranging from 0 to 9, and is a subset of a more extensive database called National Institute of Standards and Technology (NIST) [1].

**Figure 2.5:** Samples from the MNIST dataset. Digits 7, 2, 1, 0 and 4 (left to right).

Each image is already pre-processed and cropped to an equal size of 28x28, having a single digit centered to fill the image. The training and test sets are split into 60 000 and 10 000 images, respectively.

### 2.3.2 CIFAR-10

CIFAR-10 contains 60 000 tiny colored images of size 32 x 32. The training and test sets are split into 50 000 and 10 000 images, respectively.

Each image can contain an object from one of the ten non-overlapping classes; automobile, cat, horse, deer, ship, horse, bird, truck, dog, and frog. This dataset is used for image classification.

**Figure 2.6:** Samples from the CIFAR-10 dataset containing the following classes: automobile, cat, horse, deer, ship, horse, bird, truck, dog and frog

### 2.3.3 YYMNIST

Yun Yang [85] made a dataset that he called YYMNIST (Yun Yang MNIST) to do experiments quickly on object detection without using challenging datasets, such as ImageNet and COCO. YYMNIST is made out of MNIST and can be used for both classification (e.g., YYMNIST includes MNIST) and object detection tasks.

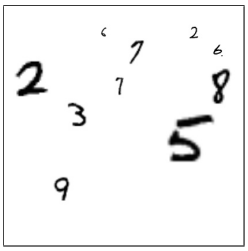


**Figure 2.7:** A sample from the YYMNIST dataset containing the following digits from left to right: 2, 9, 3, 6, 1, 7, 5, 2, 6 and 8

The GitHub repo [85] contains a script that can be used to generate a dataset containing $n$ number of images. Each generated image of size 416x416 contains 1 to $k$ number of digits. The digits are randomly placed, not rotated, non-overlapping, and in different sizes.

### 2.3.4 Apollo Synthetic Dataset

The Apollo synthetic dataset is a synthetic photo-realistic dataset that can be used for autonomous driving [74]. It provides 273 000 distinct full HD (1980x1080) video frames with high variations in weather, time of day, quality of road surface, traffic, and obstacles. These frames are collected from different areas, such as urban, downtown, residential, highway, and indoor parking garage. Different types of ground truth data are provided, such as 2D/3D object data, semantic and instance-based segmentation data, depth data, and 3D lane line data.

Looking at Figure 2.8, we can see that the dataset is structured hierarchically, starting with data type and followed by time, weather, road quality, pedestrians, traffic barriers, areas, and then finally the traffic data in its proper format depending on the data type (e.g., .jpg, .png, .txt).

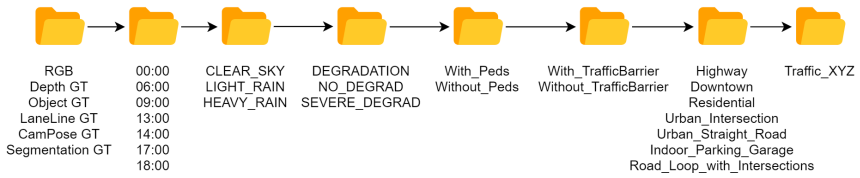
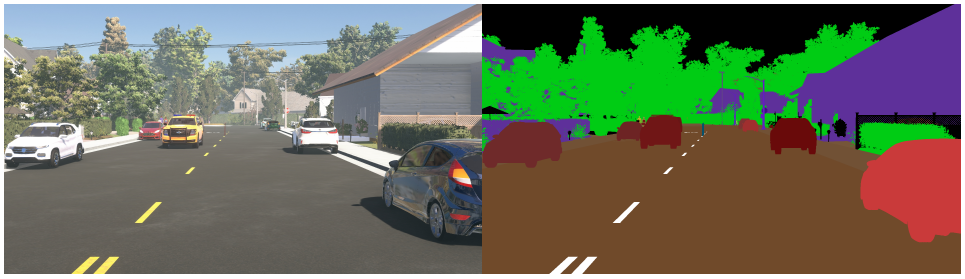



**Figure 2.8:** This is the folder structure, represented in a hierarchical fashion, of the Apollo Synthetic Dataset.



**(a)** 1920x1080 .jpg - RGB Image **(b)** 1920x1080 .png - Segmentation GT

**Figure 2.9:** A scene taken from the Apollo Synthetic dataset including its segmented ground truth.

| Thing Classes | Obj GT | Seg GT | Instance | Stuff Classes | Obj GT | Seg GT | Instance |
|---|---|---|---|---|---|---|---|
| Sedan | ✓ | ✓ | ✓ | Road | ✗ | ✓ | ✗ |
| SUV | ✓ | ✓ | ✓ | LaneMarking | ✗ | ✓ | ✗ |
| Hatchback | ✓ | ✓ | ✓ | TrafficSign | ✗ | ✓ | ✗ |
| Van | ✓ | ✓ | ✓ | TrafficLight | ✗ | ✓ | ✗ |
| PickupTruck | ✓ | ✓ | ✓ | Sidewalk | ✗ | ✓ | ✗ |
| Truck | ✓ | ✓ | ✓ | GuardRail | ✗ | ✓ | ✗ |
| Bus | ✓ | ✓ | ✓ | Sky | ✗ | ✓ | ✗ |
| Cyclist | ✓ | ✓ | ✓ | Terrain | ✗ | ✓ | ✗ |
| Motorcyclist | ✓ | ✓ | ✓ | Pole | ✗ | ✓ | ✗ |
| Pedestrian | ✓ | ✓ | ✓ | StreetLight | ✗ | ✓ | ✗ |
| TrafficCone | ✓ | ✓ | ✗ | Building | ✗ | ✓ | ✗ |
| Barricade | ✓ | ✓ | ✗ | Vegetation | ✗ | ✓ | ✗ |

**Table 2.1:** A list of thing and stuff classes that are in the Apollo Synthetic Dataset. The list shows the types of ground truths that are available for each class: "Object Ground Truth" (Obj GT) in the form of a bounding box, "Segmentation Ground Truth" (Seg GT) in the form of an image with pixel-wise segments and "Instance" whether the pixel-wise segments of this class are instance separable.

The dataset contains a total of 24 classes; 12 stuff classes and 12 thing classes. Table 2.1 gives a full overview of each class and its available ground truth for 2D bounding box, semantic segmentation and instance-based segmentation.

### 2.3.5 Waymo Open Dataset v1.0

Waymo, Google's self-driving car project, released an open driving dataset in August 2019 [80] [73]. At the time of writing this thesis, the dataset consists of thousand 20 second video segments. These include LIDAR and camera sensor data with labeled 2D/3D bounding boxes. From this dataset, only 48 video segments contain 2D bounding box annotations of three classes, vehicle, pedestrian, and cyclist. Each video segment contains a stream of approximately 200 full HD 1920x1080 frames from three cameras (e.g., front, front left, and front right). This gives us a dataset size of 29 154 samples.

New versions have been published since then: v1.0 (Aug 2019 Initial release), v1.1 (Feb 2020 Added more camera labels), and v1.2 (Mar 2020 Added new domains and test set).

**Figure 2.10:** Samples from Waymo Open taken from three different cameras (e.g., front left, front, and front right, respectively) including their object ground truths.

### 2.3.6 NAPLab's Raw Dataset (NAP-Set)

NAPLab collects data from different parts of the Trondheim area. This set contains 97 737 1280x960 images that vary in time of day and weather.

**Figure 2.11:** Samples from the NAP-Set

# Chapter 3

# Related Work

Active Learning (AL) has been studied for several years and has attracted much attention recently. Most of the earlier approaches have primarily been based on image classification [36, 16, 35, 50, 15, 32, 40, 46, 79, 21, 67]. However, in recent years some attention has grown towards using this strategy for object detection [66, 39, 9, 62, 6, 3, 4, 77, 84, 65], and, to our knowledge, a few works have also been done related to segmentation [71, 76, 51, 83, 78].

This chapter gives the reader a brief introduction to relevant research within AL, and is split into three sections: Active Learning for Image Classification, Object Detection, and Segmentation. Each section gives a broad overview of the AL approaches that have been proposed within each of these CV fields. The overview is then followed by a thorough explanation of relevant papers that our work is based on and inspired by.

## 3.1 Active Learning for Image classification

Much of the interest has been towards AL in image classification for binary classification (Jain *et al.* [35]) and the usage of uncertainty sampling, which selects the most informative samples.

Jain *et al.* [35] introduces a probabilistic variant of K-nearest neighbor that can be used with AL in multi-class scenarios. Long *et al.* [50] proposes a Gaussian Process classifier for multi-class visual recognition with multiple annotators. They use reinforcement learning to select informative samples as well as high-quality annotators, that examines the explore vs. exploit trade-off. Ducoffe *et al.* [15] presents a pool-based AL strategy using a QbC framework and batch-wise dropout to train a CNN on MNIST and USPS [75]. Holub *et al.* [32] uses a minimum expected entropy approach that selects highly informative samples that give the highest expected information gain. They demonstrate this on SVMs, Nearest Neighbour, and Kernel Nearest Neighbour classifiers. Kapoor *et al.* [40] provides a probabilistic discriminative approach for object categorization based on a Gaussian Pro-

cess regression method using a Pyramid Match Kernel. An AL strategy uses uncertainty estimates that are provided by the Gaussian Process. Li *et al.* [46] proposes an AL approach where they combine a "most uncertainty" measure with an "information density" measure, using an adaptive combination framework, to collect highly informative samples. Wang *et al.* [79] proposes a cost-effective AL Framework that can build a classifier with optimal feature representation. They update their model by progressively selecting a minority of informative samples and a majority of pseudo-labeled high confidence samples. Their approach outperforms other methods with classification accuracy, and decreases the manual human annotation effort. Gal *et al.* [21] develops an AL Framework for high dimensional data using Bayesian convolutional networks. Sener *et al.* [67] state that many of the AL heuristics in the literature are not effective with CNNs in batch setting. To solve this, they define AL as a core-set selection problem, where the goal is to find a small subset of samples from a large dataset that can be used to learn a model, to make it competitive to a model learned with the whole dataset.

### 3.1.1 Multi-Class AL for Image Classification

A pool-based multi-class AL approach, with a focus on uncertainty sampling, is proposed by Joshi *et al.* [36]. Their goal is to train a model with AL and to make it perform well using less data. Classification problems require large amounts of labeled training data to achieve high accuracy; AL wants to deal with this bottleneck.

They propose a margin-based uncertainty measure called Best-versus-Second-Best (BvSB). For each classified image, the margin between the best class probability and the second-best class probability is used as an informativeness measure to score the whole image. They state that BvSB can handle large amounts of classes and data since it is easy to compute and works without knowing the number of classes.

A Support Vector Machine (SVM) is used as the classifier, and its performance is used as evaluation. Initially, they train the classifier using a small training set containing randomly selected labeled samples. It is trained for several iterations, and during each iteration, the model selects samples from the unlabeled set based on their informativeness score, which is calculated using a specific query strategy that relies on a probability estimate from the classifier. They use pairwise coupling [81] as the method to estimate these multi-class probabilities. The samples with the highest informativeness are then labeled and added to the training set. A comparison of three different query strategies (e.g., baseline random sampler, entropy, and BvSB) is done on letter and digit recognition, object recognition on the Caltech-101 [45] dataset, and scene categorization.

Their results demonstrate that the number of training samples needed is

reduced using uncertainty sampling compared to random sampling, and that BvSB performs better than entropy by selecting more useful samples. The authors state that BvSB sampling is as good as random sampling based on the exploration of new classes, and performs much better because of its fast exploration. Entropy-based sampling performs poorly on this, and on a large number of classes. It can be a poor estimate of classification uncertainty since other unimportant classes can influence its measure (See Figures 3.1).

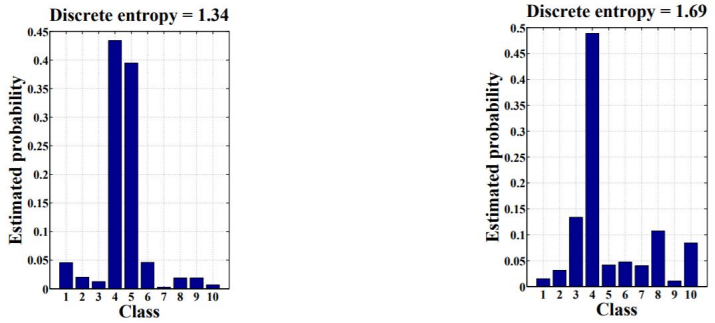


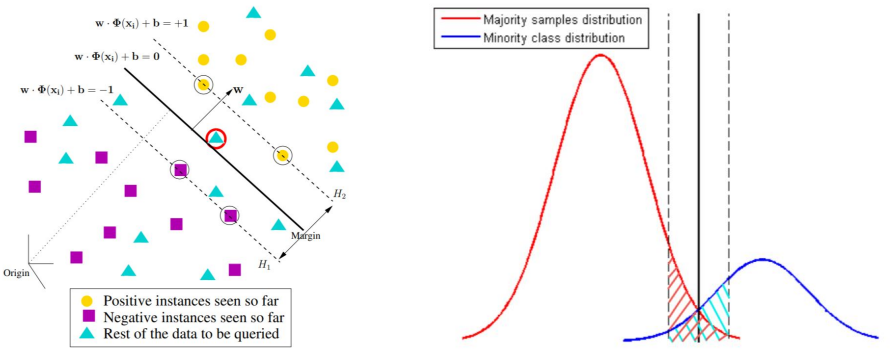

**Figure 3.1:** Joshi *et al.* [36] illustrates how entropy can be a poor estimate when measuring uncertainty. In the left figure, the classifier is highly uncertain between classes 4 and 5, and in the right figure, it is highly certain about class 4. However, in the right image the sample is still given a higher entropy score, since entropy is influenced by unimportant classes. Figures are from source: Joshi *et al.* [36].

Their margin sampling method will be part of our early iterative development. We will be using and validating BvSB on both image classification and, with various aggregation techniques, on object detection. We will achieve this by using the probability distribution over all classes (i.e., softmax vector) of each prediction.

### 3.1.2 AL in Imbalanced Data Classification

Ertekin *et al.* [16] looks into the class imbalance problem that makes the classification algorithms perform poorly. In addition, they propose an effective method to select informative samples and show that early stopping can be beneficial in AL.

A simple two-class SVM is used for the classification problem, and with the help of a hyperplane as a decision boundary, the margin between the classes is maximized. Each sample's distance to this hyperplane is measured as its informativeness score; the closer it is, the higher the informativeness (See Figure 3.2a). Samples that are closer to the hyperplane from both classes are picked, giving a more balanced set (See Figure 3.2b).




**(a)** AL with SVM **(b)** Less imbalanced data within margin

**Figure 3.2:** Figure 3.2a illustrates how informative samples (red circle) close to the hyperplane (solid line) are selected, and how labeled samples (dashed circles) create support vectors (dashed lines). Figure 3.2b shows how less imbalanced data are selected within the margin. Figures are from source: Ertekin *et al.* [16].

Problems can occur if "class imbalance" exists in a dataset. It is necessary to make sure that the AL model does not get biased towards the majority class and also make sure that the model understands the essence of the minority class. Class imbalance is when a class $A$ has a significantly low number of instances in a dataset compared to a class $B$ having a lot (Ertekin *et al.* [16]). They show that Active Learning (AL) can tackle this problem by picking fewer samples from the majority class or picking more samples from the minority class when needed.

The authors do not search the entire dataset for informative samples as in classical AL. A faster sample selection method is proposed where they pick highly informative samples from a smaller set of randomly selected unlabeled samples. This saved time and becomes easier to compute when working with large datasets.

Their results show that by changing the class ratios, AL handles class imbalance better than a random learner and is less affected. They find that early stopping increases the active learner's performance as well.

We will be using complex models for object detection and segmentation. Due to the time constraints and the complexity of these models, the proposed selection method by Ertekin *et al.* [16] will be used in this thesis, and we will be implementing early stopping to see if this can be beneficial for our AL Framework. Our focus is not on preventing class imbalance; however, we will be performing an experiment to see how different query strategies handle this issue.

## 3.2 Active Learning for Object Detection

In recent years there has been growing interest in using AL with object detection. Most works still focus on uncertainty based strategies using pool-based sampling.

Qu *et al.* [62] proposes an uncertainty based AL approach that looks at objects having low confidence scores and their regression uncertainty to declare outliers. In addition, they propose two extra weights to overcome the difficulties with class imbalance and differences in the amount of objects in images. Bietti [6] uses a linear SVM classifier for object detection on satellite images, by selecting samples that lay close to the decision boundary (i.e., simple margin method), and rely on crowd-sourcing to collect labels. Abramson *et al.* [3] uses AdaBoost [18] with AL to detect pedestrians from a moving car. Aghdam *et al.* [4] proposes a new way to score images at pixel-level to rank highly informative images that have more potential to increase the detection accuracy. Pixel-scores are aggregated to image-scores. Their approach works well on videos by avoiding redundant frames. Vijayanarasimhan *et al.* [77] presents a live learning approach, where the object detection refines its models by requesting crowd-sourced annotations from the web. Possible relevant images are gathered based on the user's text input. Yao *et al.* [84] proposes an object annotation method where the user can interactively provide annotations using the Hough Forest [22] as an object detector. Roy *et al.* [65] uses a version space reduction approach in the "difference of features" space and proposes different margin sampling approaches for SVMs.

### 3.2.1 Deep AL for Object Detection

Roy *et al.* [66] implement different AL strategies in both a black-box and white-box setting using the object detection network Single Shot Multibox Detector (SSD) [49]. In addition, they present an effective explore vs. exploit framework to achieve the best of both techniques, as discussed in Section 2.2.4.

The terms black-box and white-box gives us an indication of the type of model we are working with. In a black-box context, the model is unknown, and we can only infer based on its input and corresponding output. On the other hand, for a white-box model, the underlying architecture is known and easily understood. Put into perspective with AL, the authors make a similar assumption.

A white-box method that uses a QbC approach, is explicitly proposed for the SSD network to query informative samples. They do this by adding a set of convolutional layers on top of the multiple feature layers that makes the committee. The prediction disagreement between these layers calculates the informativeness of each prediction based on their class probabilities, and

offset to each default box in each cell.

The black-box methods are used as the baseline for comparison, and consist of various uncertainty based sampling techniques such as Minmax, Maximum Entropy, and Sum Entropy. Each sampling technique looks at the prediction probabilities for each detection in an image. Minmax selects a detection having the highest class probability and sets it as the image score. Images having the lowest class probability scores are selected. Maximum Entropy and Sum Entropy calculates the entropy of each detection and sets the highest entropy and sum of entropy as the image score, respectively.

A model is initially trained using 10% of the PASCAL-VOC dataset, and is used as a starting point for the active learners. Each black-box and the white-box method is then re-trained by adding batches of actively selected samples.

Their results show that a specific version of their white-box method, which uses class probabilities to calculate the image margin, outperforms the other white-box methods. All their white-box methods outperform the black-box baseline methods. In the baseline methods, Sum Entropy performs better than Maximum Entropy due to its way of selecting images with more objects. They state that their results improve by adding exploitation.

Their proposed approach was part of our early research when working with AL. Our primary takeaways were the two black-box query strategies, maximum entropy and sum entropy. It is worth to mention that some time was put into reproducing their results using the white-box method with SSD. However, due to insufficient documentation and some unanswered questions, this attempt was put on ice. Nevertheless, their explanation of these black-box methods strengthened our initial understanding of how query strategies can be implemented in object detection using aggregations.

### 3.2.2 Localization-Aware AL for Object Detection

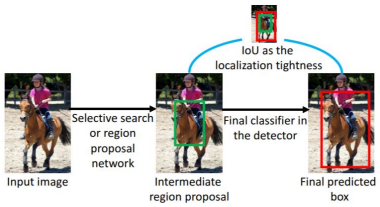
Kao *et al.* [39] presents two metrics that measure the prediction localization uncertainty, "localization tightness," and "localization stability." The former measures the overlapping ratio between the region proposal and the final prediction (refined from the proposal), and the latter measures the variation of object location when noise corrupts the input images. Two object detectors use these metrics; Faster R-CNN uses both, and SSD uses only "localization stability."

Prediction uncertainty is measured using both classification and localization uncertainty. For each bounding box in an image, the classification uncertainty is denoted as $1 - max(softmax\ vector)$ (i.e., one minus the highest class probability). The classification uncertainty of an image is the highest bounding box uncertainty from all detections in that image.

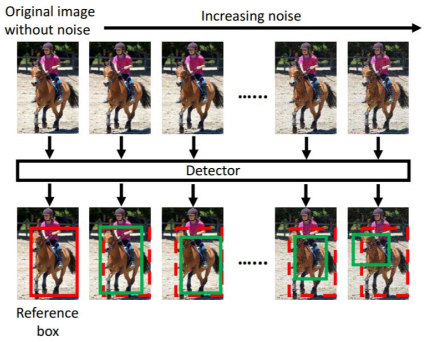
Localization tightness is measured by calculating the Intersection over Union (IoU) between the region proposal, given by a Region Proposal Net-

work (RPN) or selective search, and the refined bounding box (See Figure 3.3a). The authors argue that this is a reasonable estimate as, if a region proposal does not need refinement, it has a high certainty that there exists an object instance; otherwise, it does not. Images with inconsistency between classification and localization are seen as highly uncertain, and are selected.

Localization stability is measured by giving the object detector a set of images with increasing noise; inference is run multiple times with the same image set (See Figure 3.3b). If the results vary, the image might be highly informative to be labeled. The uncertainty is measured by calculating the IoU between the original bounding box prediction, of an image with no noise, and a set of corresponding predicted bounding boxes, from images with varying amounts of noise.




**(a)** Localization Tightness




**(b)** Localization Stability

**Figure 3.3:** Figure 3.3a illustrates how localization tightness is measured by calculating the IoU between the region proposal and the refined bounding box. Figure 3.3b illustrates how localization stability is measured by calculating the change in predictions by taking the IoU between the predicted boxes (green) and the reference box (dashed red) with increasing noise. Figures are from source: Kao *et al.* [39].

They run experiments using various combinations of their metrics localization tightness (LT), localization stability (LS), and classification uncertainty (C). Their results demonstrate that LT and LS has a positive impact on AL and perform better than only using C, and that they achieve a similar accuracy using less labeled data.

We attempted to implement their localization measures using Matterport's Mask R-CNN framework [2]. Unfortunately, this measure was not used, due to framework issues and the change to the up-to-date platform Detectron2 (See Appendix C.1 for more info). However, a similar approach of calculating the IoU between predictions (e.g., bounding boxes and masks) was implemented alongside the dropout technique proposed by Morrison *et al.* [54] for segmentation, which is explained in Section 3.3.1.

### 3.2.3 AL for Deep Object Detection

Brust *et al.* [9] combines AL on object detection with incremental learning that enables continuous exploration. In this scenario, new data and classes are added to the dataset over time. Furthermore, they propose various aggregation metrics and query strategies for object detection models, and present an approach to better handle class imbalances under sample selection.

As a query strategy, they propose 1-vs-2, which is a margin sampler. 1-vs-2 measures the margin between the highest and the second-highest class probability, similar to BvSB (Joshi *et al.* [36]), and expects an output of class probabilities for each detection (e.g., softmax vector). In order to aggregate the scores from detections, they propose the aggregation techniques sum, average, and maximum (Section 2.2.3 explains these in detail).

The class imbalance problem is tackled by weighting instances in an image and selecting those who are predicted as a minority class that is underrepresented in the training set compared to majority classes. For each instance, this metric is multiplied with the margin metric 1-vs-2 before aggregation.

Two experiments are conducted using a random learner as the baseline, and on the PASCAL VOC 2012 dataset. First, by using AL on object detection with continuous exploration, and then, by including their proposed weighting metric, which prevents class imbalance.

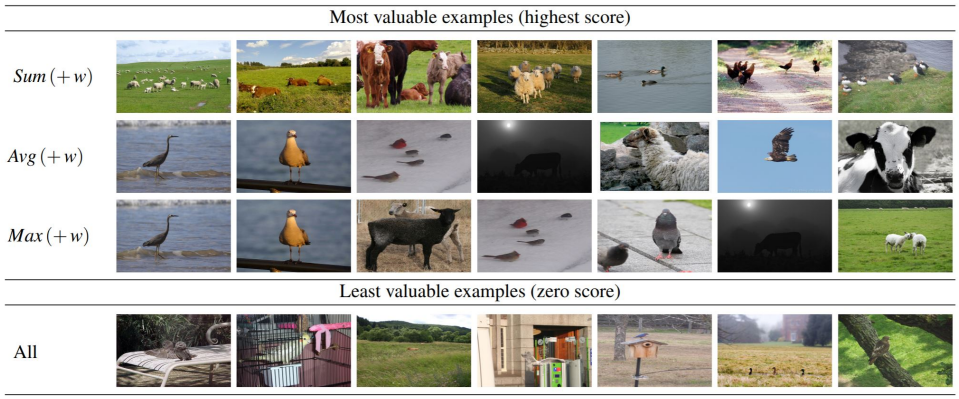



**Figure 3.4:** Brust *et al.* [9] presents selected samples from using the aggregation techniques sum, average, and maximum, including a weight metric. Figure source: Brust *et al.* [9].

They state that by adding new classes and new samples to the training set causes minimal loss to what the learner already knows, and their results demonstrate that incremental learning works. In addition, all aggregation techniques are able to perform better than the random learner. Sum is the best performing, due to its way of selecting samples containing more

instances. Maximum and average performs similarly due to most samples containing a single object. As for using the weighting metric, they notice a performance improvement for each active learner, especially for sum that selects instance-rich samples that might contain rare instances. Figure 3.4 shows the selected samples.

Our experiments will be using the aggregation techniques sum, maximum, and average, but in combination with different query strategies. We will be examining their results to see if we notice any similar performance patterns using these aggregation techniques.

## 3.3 Active Learning for Segmentation

To our knowledge, there has not been much interest in using AL with segmentation as compared to object detection and image classification.

Vezhnevets *et al.* [76] proposes an "expected change"-based AL approach for semantic segmentation using a pairwise Conditional Random Field. Mackowiak *et al.* [51] proposes a cost-effective AL Framework for multi-class semantic segmentation. The authors want to find a minimal set of highly informative samples while minimizing the human annotation effort. Yang *et al.* [83] presents an AL Framework for FCN that reduces the annotation effort by giving suggestions on annotation areas that will be most effective. Vijayanarasimhan *et al.* [78] presents an AL Framework that can predict the effort and information gain trade-off. This framework ranks unlabeled and partially labeled data based on how much they are worth to select informative samples.

### 3.3.1 Uncertainty-aware Instance Segmentation using Dropout Sampling

Morrison *et al.* [54] addresses the task of instance-based segmentation by looking at both spatial and semantic uncertainty of a prediction using dropout sampling. Their goal was not to perform AL using this uncertainty measure or to make an AL Framework. However, their proposed approach might be highly useful in order to measure the informativeness of segmentations.

By adding dropout layers to the fully-connected layers of Mask R-CNN, they run inference over the same image multiple times in order to measure the segmentation uncertainty based on predictions. This approach can be seen as a committee of models where their disagreement is measured.

Inference is run over an image 16 times, and the predictions are grouped, using a Basic Sequential Algorithmic Scheme [53], into observations based on their mask IoU (See Figure 3.5). Each observation in an image contains a set of predictions that include softmax vectors, bounding boxes, and

segmentation masks. A mean of the predictions is calculated for each observation.

Semantic uncertainty is measured as the highest class probability of the mean softmax of each observation. Spatial uncertainty is measured as the mean IoU between the mean mask and every other mask in each observation. A third appearance-measure is calculated by counting the number of times a prediction appears out of 16 inferences. The three uncertainty measures are combined in order to get a weighted hybrid uncertainty score for a single observation.

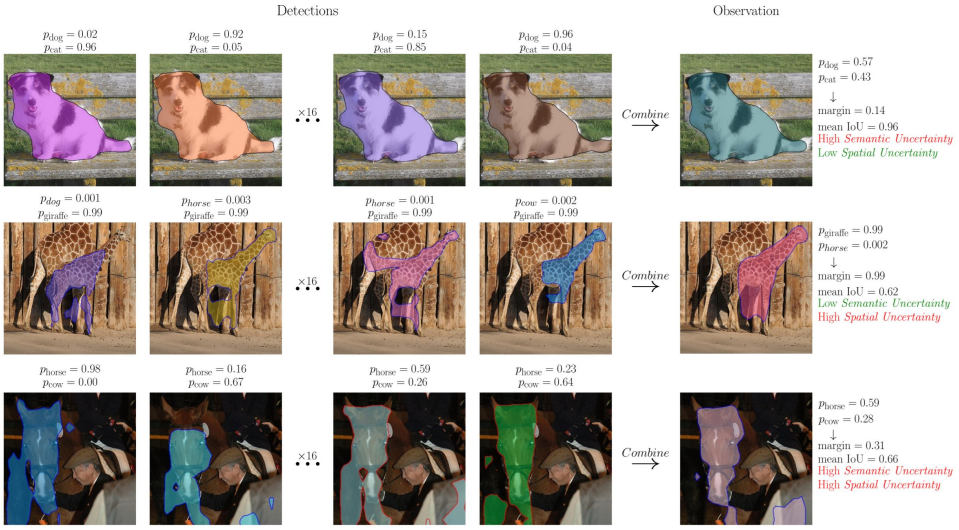



**Figure 3.5:** Morrison *et al.* [54] illustrates how semantic and spatial uncertainty is measured by running inference over images 16 times with dropout layers. **Top:** The segment is classified as a dog or cat inconsistently (i.e., high semantic uncertainty). The segment mask is consistent (i.e., low spatial uncertainty). **Mid:** The segment is classified as a giraffe consistently (low spatial uncertainty). The segment mask is inconsistent (high spatial uncertainty). **Bottom:** The segment has both high semantic and spatial uncertainty. Figure source: Morrison *et al.* [54]

To be clear, the authors do not evaluate their approach using AL with any baseline. It is, therefore, hard to say if this method works well in this setting. They evaluate their approach by using a pairwise Probability-based Detection Quality metric (PDQ) [29]. This metric computes a PDQ score based on spatial and semantic probabilities. They perform a grid search on their uncertainty metrics to find good thresholds. Highly informative detections that are below these thresholds are neglected as they might be false positives. Since PDQ is affected by false positives, removing these improved their evaluation scores.

We will be using a similar query strategy with a QbC framework, which

will be heavily based on their method. In addition, we will add an extra measure that considers the spatial uncertainty of the predicted bounding boxes. Furthermore, we will evaluate the query strategy's performance and compare it to other query strategies. We will not be focusing on finding optimal thresholds to neglect false positives, as these will be chosen arbitrarily.

### 3.3.2 AL for Road Segmentation using CNN

Sörsäter [71] proposes a QbC AL approach using Monte Carlo dropout to perform semantic segmentation on roads. Monte Carlo dropout is used to measure a model's uncertainty similar to Gal *et al.* [20]. The authors use Venoeer's custom ENet [58] as their CNN and their own dataset, which contains 54 000 images having the three segment classes road, lane and void.

Query strategies such as least confident, margin sampling, entropy, and Monte Carlo dropout are being used and compared with a baseline random learner. Their Monte Carlo dropout approach runs inference over every sample $N$ time, and collect the predictions; however, the uncertainty is calculated differently compared to Morrison *et al.* [54].

Image uncertainty is calculated in a pixel-wise manner by measuring the uncertainty of each pixel prediction using their softmax vector, and aggregating each pixel uncertainty by taking the mean over all pixels uncertainties.

Their results demonstrate that all query strategies perform better than the baseline random learner, and that the entropy learner is the best performing. However, the dropout learner does not perform any better than the entropy learner.

Our focus will be on implementing the technique mentioned by Morrison *et al.* [54] and not the techniques presented in this paper. We will rather make an indirect comparison to examine how a query strategy using dropout performs. However, we believe the work presented in this paper is worth mentioning as it is highly relevant in an autonomous setting.

# Chapter 4

# Methodology

In this chapter, we will start by presenting the Active Learning Framework (AL Framework). Furthermore, we will list the AL query strategies and aggregation techniques for each CV task. Finally, we present the experiment structure, AL process, and evaluation details followed by the experimental setup.

Our AL Framework is considered as a black-box problem that only relies on the inputs and outputs and is developed iteratively. Therefore, some implementation choices are made based on findings from previous research and experiments, while others are purely made from try and fail approaches. These will be briefly justified in this chapter and in more detail throughout Chapter 5. Our focus is on exploring and reviewing AL approaches to find a small set of highly informative training samples; therefore, we do not focus on training and model optimization to achieve the "best" possible model.

## 4.1 AL Framework

We aim to apply AL on both object detection and segmentation in an autonomous setting. Therefore, we need a good foundation of high-performance state-of-the-art models. The object detection library Detectron2 [82] is a good fit, due to its broad usage, state-of-the-art results, and well-documented open-source code. It has all the necessary state-of-the-art models (e.g., Faster R-CNN and Mask R-CNN), while being fast, stable, highly customizable, and up-to-date.

We implement a modular pool-based AL Framework to be used with Detectron2 [82]. The framework is mainly an uncertainty based framework, but it has the ability to resemble a QbC framework virtually. An uncertainty sampling framework can be seen as a black-box problem that focuses on using black-box query strategies. An advantage of using these strategies is that they measure the sample informativeness by considering the model's inputs and outputs. Thereby making them highly generalizable for other

architectures of choice and easily usable for further development. For further details about the architecture of the AL Framework and to download our code, see Appendix A.1.

Our AL Framework can use new models and datasets as long as they are implemented following Detectron2's documentation, and uncertainty based query strategies can easily be added and implemented as long as they handle the expected input and give an expected output.

The AL Framework runs an experiment using all the provided learners in a sequential fashion where each learner is run for a number of Active Learning Iterations (ALIs). A single ALI consists of: initialization and configuration, training, evaluation, inference and prediction, logging, and clean-up. Figure 4.1 illustrates the workflow of our AL Framework.

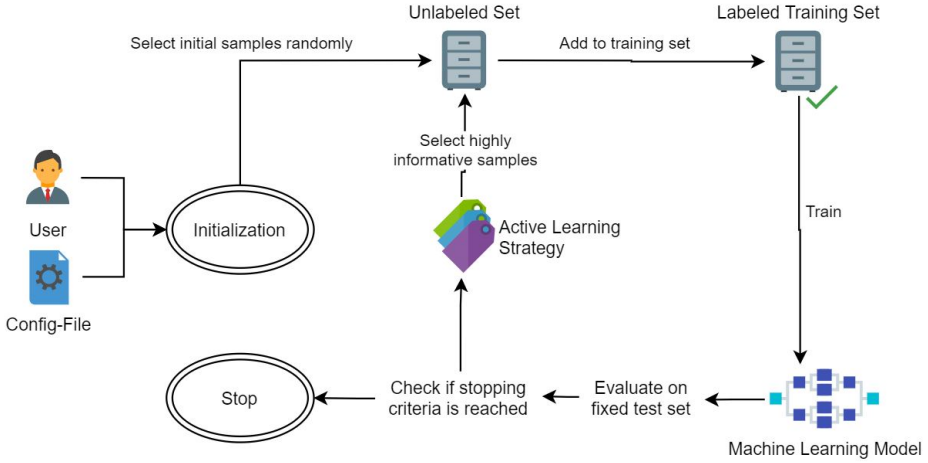



**Figure 4.1:** This figure illustrates the AL Framework workflow. A user provides a config-file and initializes the framework. Initially, samples are selected randomly, labeled, and added to the training set; labels of samples are obtained from a dataset and not manually annotated. Then, a model is fully trained, evaluated on a test set, and used with AL to select a new set of highly informative samples that are added to the training set. This process is repeated for a number of ALIs.

The user can provide a configuration file to run a custom experiment. The following list is part of our contributions and gives a quick overview of the supported functionality:

- **Active Learning:** The user can provide the number of Active Learning Iterations to be run, the number of samples to be selected each Active Learning Iteration, a list of query strategies to be used, size of the initial training, validation (used with early stopping), and test set,

and the number of samples to be selected during inference from the unlabeled set.

- **Whole Spectrum:** The user can choose whether samples should be selected from the whole informativeness spectrum or only from the top of the spectrum (e.g., top highly informative samples). If the whole uncertainty spectrum is selected, the user has to provide a sample ratio, which sets the thresholds.
- **Early Stopping:** The user can activate early stopping, by providing parameters such as patience, delta, and the evaluation period. With early stopping, the model is trained for much longer, and early stopping is initiated based on the Average Precision on the validation set.
- **Data Diversity**: The user can ensure data diversity. Highly similar samples are removed from the scored sample set using the similarity metric LPIPS (Zhang *et al.* [86]).
- **Model and Weights:** The user can select the type of model to be used (e.g., Faster R-CNN or Mask R-CNN) and what weights to be loaded from Detectron2's model zoo, or from a previous experiment.
- **Dataset and Outputs:** The user can provide which dataset to be used and the output directory for the experiment results and logs. Currently supported datasets: YYMNIST, Apollo Synthetic, Waymo Open and Raw-NAPLab.

The AL Framework logs and saves several values during each ALI that is used for comparison and illustrative purposes in Chapter 5:

- Performance values on the test set following COCO's detection evaluation metrics [11]. See Appendix A.3 for more details.
- Total instance count of the training set, and for each class.
- Top 10 samples with high and low informativeness.

#### Changes to Detectron2

Detectron2 is a fairly new but stable object detection platform that was released in late 2019 and is still being frequently updated. Some functionality is still to be implemented and is not included out of the box. The following functionality is added to make Detectron2 compatible with our framework:

- During inference, prediction scores in the form of a softmax vector are obtained from each detection in a sample.
- A custom head using dropout layers is added, which is used by the query strategy DROPOUT during inference.

- A custom training logic loop is implemented to be used with early stopping and evaluation on the validation set.
- A general DatasetEvaluator is implemented and used for model evaluation on the supported datasets.
- A custom DataLoader is implemented for each supported dataset.

## 4.2 AL Strategies

We implement several query strategies for each CV task (e.g., image classification, object detection, and instance-based segmentation) during our iterative development. Our proposed query strategies and aggregation techniques are inherited from proposals and findings in previous work and research.

This section introduces our learners, which either use a query strategy alone, or combine it with an aggregation technique (i.e., a learner using Sum as aggregation technique and Entropy as query strategy is defined as a Sum-Entropy learner). Throughout this chapter and the following chapters, we will use unique abbreviations for each learner (e.g., Sum-Entropy learner (SUMENT)).

### 4.2.1 Learners

#### Baseline Learner

- **Random (RAND)** - A learner that selects samples randomly from the unlabeled set. All experiments use this learner as baseline for comparison.

Using a pool-based sampling technique, the learner has to search through the whole unlabeled set to find informative samples. As this process is repeated for every ALI, it can become computationally intensive if the unlabeled set is large. We use the faster sample selection method proposed by Ertekin *et al.* [16]. Our learners use this method to search through a smaller set of randomly selected samples from the unlabeled set, to then find informative samples.

The following learners require a probability distribution over all classes in the form of a softmax vector (i.e., prediction probabilities should sum up to one):

#### Image Classification Learners

Since an image consists of a single detection, aggregation techniques are not needed to score the entire image. We use the query strategy Best-versus-Second-Best (BvSB) proposed by Joshi *et al.* [36].

- **Entropy (ENT)** - Calculates the entropy of the softmax vector and sets it as the image score. Images having a high entropy score have high informativeness and represent high uncertainty with respect to the model.
- **Best-vs-Second-Best (BVSB)** [36] - Calculates the margin between the best prediction and the second-best prediction in the softmax vector and sets $1 - margin$ as the image score. Images having a high score have high informativeness and represent high uncertainty with respect to the model.

The learners ENT and BVSB give a score between 0 and 1. However, for ENT, a score closer to 1 indicates high informativeness, while for BVSB, a score closer to 0 indicates high informativeness. For implementation purposes, using ENT, each detection in an image is scored as (score1, score2, ..., scoreN), while using BVSB, each detection is scored as (1-score1, 1-score2, ..., 1-scoreN). By doing so, both learners will indicate high informativeness if their scores get closer to 1, and both learners will be able to use the aggregation techniques presented in the following sub-section.

### Object Detection Learners

Since an image can contain multiple detections with different scores, aggregation techniques are needed to score the entire image. We use the aggregation techniques presented by Brust *et al.* [9].

- **Sum (SUM)** - Takes the sum of all scores. Prefers images containing many informative detections.
- **Maximum (MAX)** - Takes the maximum of all scores. Prefers images containing a single highly informative detection, and is not affected by the number of detections.
- **Average (AVG)** - Takes the average of all score. Can prefer images containing many informative samples as much as images containing a single informative sample.

Images having no detections are not given an image score and are not counted as informative samples. We assume that this might be a case earlier in the ALIs. However, over time, the learners get better at detecting objects. Our focus is on giving the learner what it is uncertain of with respect to what it detects.

Each query strategy is generalized and expects a specific input to give the desired output. The input should contain a set of $n$ images $\{img_1, ..., img_n\}$ where each image should contain a set of $m$ detections $\{d_1, ..., d_m\}$. If the reader is curious, pseudo-codes of these learners are provided in Appendix B.

- **Sum Entropy (SUMENT)** - Calculates the entropy of the softmax vector and sets it as the score for each detection in an image. The **sum of detection scores** in an image will be set as the image score.
- **Max Entropy (MAXENT)** - Similar calculation as SUMENT, but the **highest scoring detection** in an image will be set as the image score.
- **Average Entropy (AVGENT)** - Similar calculation as SUMENT, but the **average of the detection scores** in an image will be set as the image score.
- **Sum of Best-vs-Second-Best (SUMBVSB)** - Calculates the margin between the best prediction and the second-best prediction in the softmax vector and sets $1 - margin$ as score for each detection in an image. The **sum of detection scores** in an image will be set as the image score.
- **Max of Best-vs-Second-Best (MAXBVSB)** - Similar calculation as SUMBVSB, but the **highest scoring detection** in an image will be set as the image score.
- **Average of Best-vs-Second-Best (AVGBVSB)** - Similar calculation as SUMBVSB, but the **average of the detection scores** in an image will be set as the image score.

#### Instance-Based Segmentation Learners

Aggregation techniques are needed to score an image that might contain multiple segments. Here, we only use the aggregation technique **Sum (SUM)** with the Dropout (DROPOUT) learner.

This query strategy expects a specific input to give the desired output. Since inference is run over the same image set $k$ times using dropout, the input set should contain $n$ images $\{img_1, ..., img_n\}$ having $k$ prediction sets each. Each prediction set in each image $img_n$ should contain a set of $m$ detections $\{d_1, ..., d_m\}$.

- **Dropout (DROPOUT)** - By following Morrison's *et al.* [54] uncertainty measuring method, we add dropout layers in the final layers of the model and use these only during inference (e.g., Monte Carlo dropout). We run inference over the same image multiple times and generate object observations by adding overlapping detections together. For each observation, we calculate the mean softmax, mean bbox and mean mask, and use these values to calculate the semantic uncertainty, spatial bounding box uncertainty, spatial mask uncertainty, and number of appearances of each detection out of number of

inferences (See Section 3.3.1 for details). Take notice, we add a fourth uncertainty (i.e., spatial bounding box uncertainty), since we believe that the disagreement in bounding box prediction should be accounted for to get an overall uncertainty.

The informativeness of an image is calculated as the sum of detection scores, where each detection is given a final weighted uncertainty score as *semantic uncertainty* $*$ *spatial bbox uncertainty* $*$ *spatial mask uncertainty* $*$ *appearances* (e.g., *appearances* - number of appearances out of number of inferences). The informativeness score can vary between 0 and 1 for each detection; detections having a final score under a threshold are ignored as they often consist of false positives, as stated by Morrison *et al.* [54]. This threshold can be used with the various uncertainty metrics. We use a threshold of 0.2 on the final weighted uncertainty score (i.e., detections having a lower score than 0.2 are ignored). A pseudo-code of this algorithm is provided in Appendix B.

## 4.3 Experiment Structure

As part of our implementation plan, and in order to implement a general AL Framework that fits our purpose, we examine four vital components. This structure is important to reflect upon and can be used as a guide for creating a good foundation to perform AL experiments:

**Computer Vision Task:** Choose a CV task that you want to work with (e.g., Image Classification, Object Detection, Segmentation).

**Model:** Select or build an applicable model that can be trained to perform these tasks.

**Dataset:** Find or assemble a relevant dataset that fits the chosen task.

**Active Learning Strategy:** Build an active learning strategy by choosing a type of query scenario, query strategy framework, types of query strategies, and aggregation techniques.

### 4.3.1 Active Learning Process

Each experiment follow a general AL flow, which lasts for a number of Active Learning Iterations (ALIs). Initially, samples are selected randomly from an unlabeled set to create an initial training set and to train an initial model. The learners then use this initial model as a starting point. During each ALI, a learner selects a new set of informative samples from the unlabeled set, adds it to its training set, does a full training, and evaluates itself on a test set. A learner is not trained from scratch each ALI; it uses the weights

from a previous ALI. Figure 4.2 gives a simple illustration of how data grows during the AL process.

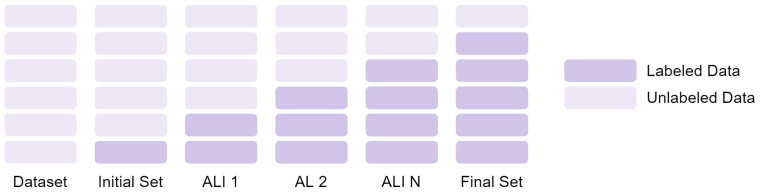



**Figure 4.2:** This figure illustrates how training data grows during each Active Learning Iteration using Pool-Based Sampling

### 4.3.2 Evaluation Details

Each experiment goes through the same evaluation process. A set of active learners are compared to a baseline random learner to see if they can reach a similar performance using a smaller but highly informative training set.

For image classification, the performance is measured as the prediction accuracy on a test set. For object detection and segmentation, our implemented DatasetEvaluator uses Detectron2's default evaluations process, which returns several Average Precision (AP) metrics. Here, the performance is measured as AP50 on a test set (See Appendix A.3 for more details).

The performance of the learners can vary a lot based on the initial set. Therefore, we run some experiments for a maximum of three times, using different initial training sets, to make sure the results are reliable and to get an average. This is only done on experiments that are less computationally heavy. The averaged results for each learner are shown in a graph as the performance over training set size. We measure a learner's performance on a test set after each ALI.

# Chapter 5

# Experiments and Results

This chapter presents the experiments, results of the experiments, and discusses the results if needed. As part of our iterative development, doing smaller experiments helps us understand Active Learning (AL), and is needed to build a good foundation. Therefore, findings and results from one experiment impacts the subsequent experiments. Keep in mind that we set the experiments realistically up in order to imitate a real-world scenario, which includes limited data resources, unevenly distributed data, and limited time.

We present each experiment by giving a problem description, followed by a technical description of the dataset, the experimental setup, and the AL setup. In addition, we explain extra functionality, if added. Finally, we present and discuss the results.

### Experiments Overview

Each experiment uses a pool-based sampling technique, a RAND learner as baseline for comparison, and can consist of multiple sub-experiments. We follow the four components explained in Section 4.3 to structure each experiment. The experiments are listed below with increasing complexity:

| **EXP 1** | **A Simple "Hello World"-Experiment with AL** | |
|---|---|---|
| CV Task: | Image Classification | AL Strategy: |
| Model: | Simple CNN, modAL | Uncertainty Based Framework |
| Dataset: | MNIST | Query Strategies: ENT, BVSB |
| Sub-Experiments | | |
| EXP 1.1 | More ALIs | Identical Setup as EXP 1 |
| EXP 1.2 | Using CIFAR-10 | Different from EXP 1<br>Dataset: CIFAR-10 |

| **EXP 2** | **Using a Complex Framework** | |
|---|---|---|
| CV Task: | Object Detection | AL Strategy: |
| Model: | Faster R-CNN, Detectron2 | Uncertainty Based Framework |
| Dataset: | YYMNIST | Query Strategies: |
| | | MAXENT, SUMENT, AVGENT |
| Sub-Experiments | | |
| EXP 2.1 | All Margin Learners | Different from EXP 2 |
| | | Query Strategies: SUMENT, |
| | | MAXBVSB, SUMBVSB, AVGBVSB |

| **EXP 3** | **Using a Synthetic Self-Driving Dataset** | |
|---|---|---|
| CV Task: | Object Detection | AL Strategy: |
| Model: | Faster R-CNN, Detectron2 | Uncertainty Based Framework |
| Dataset: | Apollo Synthetic | Query Strategies: |
| | | MAXENT, SUMENT, AVGENT |
| Sub-Experiments | | |
| EXP 3.1 | Spectrum vs. | Different from EXP 3 |
| | No-Spectrum | Query Strategies: |
| | | MAXENT, SUMENT, AVGENT |
| EXP 3.2 | Unbalanced | Different from EXP 3 |
| | Initial Set | Query Strategies: MAXENT, |
| | | SUMENT, MAXBVSB, SUMBVSB |

| **EXP 4** | **Using a QbC Framework** | |
|---|---|---|
| CV Task: | Object Detection | AL Strategy: |
| | Instance-Based Segmentation | QbC Framework |
| Model: | Mask R-CNN, Detectron2 | Query Strategies: MAXENT, |
| Dataset: | Apollo Synthetic | SUMENT, SUMBVSB, DROPOUT |

| **EXP 5** | **Moving towards a Real-Life Dataset** | |
|---|---|---|
| CV Task: | Object Detection | AL Strategy: |
| | | Uncertainty Based Framework |
| Model: | Faster R-CNN, Detectron2 | Query Strategies: |
| Dataset: | Waymo Open | MAXENT, SUMENT, |
| | | AVGENT, MAXBVSB, SUMBVSB |
| Sub-Experiments | | |
| EXP 5.1 | Ensuring Data Diversity | Different from EXP 5 |
| | | Query Strategies: MAXENT, |
| | | SUMENT, MAXBVSB, SUMBVSB |
| EXP 5.2 | Introducing Early Stopping | Different from EXP 5 |
| | | Query Strategies: |
| | | MAXENT, SUMENT |

| **EXP 6** | **Finding Informative Samples** | |
|---|---|---|
| CV Task: | Object Detection | AL Strategy: |
| | | Uncertainty Based Framework |
| Model: | Faster R-CNN, Detectron2 | Query Strategies: DROPOUT, |
| Dataset: | NAPLab's Raw Data | MAXENT, SUMENT, AVGENT |

## 5.1 EXP 1 - Simple Active Learning

**A Simple "Hello World"-Experiment with AL**
**Task:** Image Classification
**Model:** Keras' Simple CNN with modAL
**Dataset:** MNIST

### 5.1.1 Problem Description

To understand the fundamentals of AL, we implement an AL Framework using modAL [12] to quickly validate the usage of this strategy, and to gain useful information and insight on how we can later proceed towards more complex frameworks. ModAL [12] is a modular active learning framework for python3. We use a simple CNN from Keras [41] on the MNIST dataset [44] to perform image classification. Keras' model can achieve an accuracy of 99.25% when trained on the entire dataset for 12 epochs. We train the CNN from scratch for a number of ALIs using the learners RAND, ENT, and BVSB (Joshi *et al.* [36]).

### 5.1.2 Dataset

We use MNIST and split it into a training, test, and unlabeled set. The initial training set is balanced and contains two images from each digit, giving it a size of 20 images. The test set contains 10 000 images and is fixed throughout the whole experiment. The remaining images virtually create an unlabeled set of 59 980 images.

### 5.1.3 Setup

We use the AL Framework modAL [12], which is highly modifiable and allows us to create an AL workflow quickly. We implement and add the learners, RAND, ENT, and BVSB, to the framework. RAND selects $n$ samples randomly. ENT and BVSB select 2000 random samples from the unlabeled set (i.e., fast selection method as stated in Section 4.2.1), run inference over them, calculate their informativeness based on their predictions, and return a set of $n$ top samples having the highest informativeness score.

#### ALIs

We repeat this process for 50 ALIs. During each ALI, we train a model for 50 epochs using a learning rate of 0.001. We initially train a model using the balanced training set, which contains 20 images. Then, we use the initial model as a starting point to train the learners RAND, ENT, and BVSB sequentially. Each learner adds a set of 10 images to its training set based on their informativeness score.

### 5.1.4 Results

We run the experiment three times using different initial sets to get an average. Figure 5.1 shows the average test accuracy for each learner.

All learners are able to quickly reach an accuracy of 85% within the 14 first ALIs. The best-performing learner is BVSB, followed by ENT. The baseline RAND learner reaches its top accuracy of 93.29% using 520 randomly selected training samples (ALI 50). ENT having a top accuracy of 96.21% reaches the same accuracy as RAND using 290 training samples (ALI 27). However, BVSB having the top accuracy of 97.11%, reaches the same accuracy as RAND using only 240 training samples (ALI 20). The active learners BVSB and ENT outperform the baseline RAND learner by a good margin.

Selected images are saved during each ALI for ENT and BVSB, and are shown in Figure 5.2 and 5.3, respectively. We notice a clear difference in hard and easy images selected by both active learners.

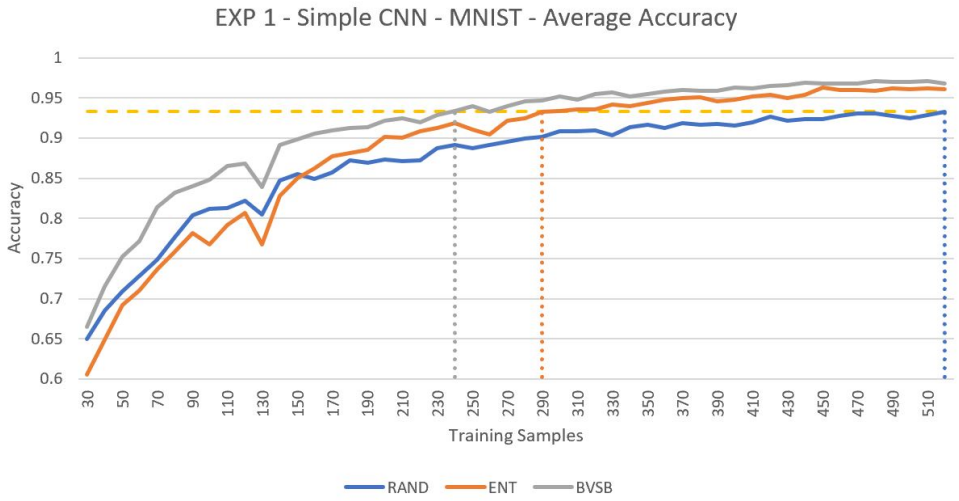



**Figure 5.1:** Performance results from EXP 1 using AL with Image Classification on MNIST. The accuracy is calculated using the test set of 10,000 samples, and averaged over three independent runs from ALI 1 to 50.

**(a)** Digits 9, 4, 5, 7, 2, 7, 4, 9, 8 and 7 **(b)** Digits 7, 5, 7, 6, 7, 3, 7, 7, 7 and 7

**Figure 5.2:** Images selected by ENT, from the MNIST dataset, at ALI 48. **Left:** Top 10 hard images with high informativeness scores. **Right:** Top 10 easy images with low informativeness scores.

**(a)** Digits 3, 4, 2, 6, 9, 3, 4, 5, 8 and 1 **(b)** Digits 3, 5, 8, 8, 2, 2, 2, 3, 8 and 8

**Figure 5.3:** Images selected by BVSB, from the MNIST dataset, at ALI 48. **Left:** Top 10 hard images with high informativeness scores. **Right:** Top 10 easy images with low informativeness scores.

This particular experiment confirms that the active learners can perform better than a baseline learner, and can reach a higher accuracy using a smaller training set. We obtain promising results that are comparable with the performance patterns seen in Joshi *et al.* [36]. The performance graph shows that BVSB starts to select informative samples early in the ALIs compared to ENT, which is why it reaches a high accuracy faster than both ENT and RAND. ENT struggles to reach a higher accuracy than RAND in the first few ALIs; this is mostly due to it not having understood the underlying structure of the data by favoring samples from specific classes. Joshi *et al.* [36] states that entropy-based selection is poor at exploring compared to BVSB and RAND, as explained in Section 3.1.1.

From the first run, we compare the class distribution of each learner by counting the number of instances from each class in the training set. Figure 5.4 shows the class distribution from ALI 50. RAND ends up with a more balanced distribution compared to the active learners who focus on selecting more instances of digits that they are most uncertain of (e.g., digit 8 and 9).

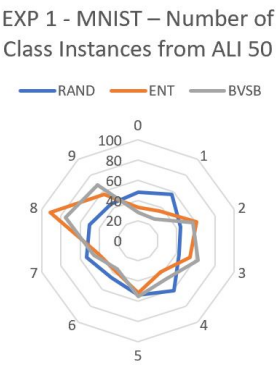



**Figure 5.4:** This plot illustrates the number of class instances (digits 0-9) in training set for each learner at ALI 50.

### 5.1.5 EXP 1.1 - More Active Learning Iterations

We rerun the experiment to test if the learners achieve similar performance if we train them for a more extended period with an increasing training set. The total number of ALIs is increased from 50 to 200, thereby increasing the training set size.

Previously, the baseline RAND learner achieves an accuracy of 93.29% at ALI 50 using 520 samples. By looking at Figure 5.5, we see that RAND is struggling to reach an accuracy above 97% while ENT and BVSB are getting closer to 99%. BVSB manages to reach this using only 3.26% of the samples from the entire MNIST training set (1960 samples). We observe that ENT and BVSB starts to reach a similar accuracy after 100 ALIs. It is hard to say whether RAND will reach their accuracy if it is trained long enough.

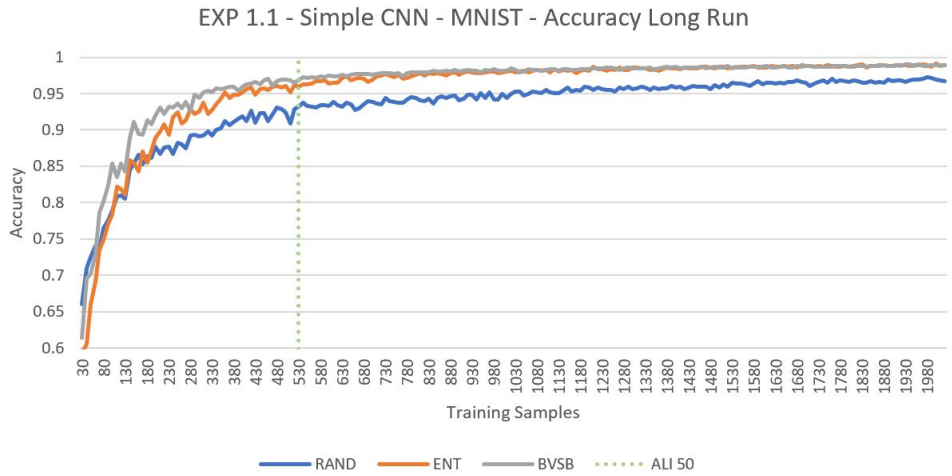



**Figure 5.5:** Performance results from EXP 1.1 by running AL with Image Classification on MNIST, for a longer period of time. The accuracy is calculated using the test set of 10,000 samples, from a single run, and from ALI 1 to 200.

### 5.1.6 EXP 1.2 - Using the CIFAR-10 Dataset

We rerun the experiment using the dataset CIFAR-10, due to it being more complex than MNIST. Some changes are made to the setup from EXP 1: We use a different CNN for this task, we increase the epochs to 150, decrease the batch size to 64, increase the initial set size to 500 samples, increase the number of ALIs to 100 and we let the learners add 250 samples to the training set each ALI.

We run this sub-experiment three times and take the average test accuracy of each learner. The results are shown in Figure 5.6. All learners reach an accuracy of 70% within the first 30 ALIs using 8000 samples. In the subsequent ALIs, both active learners outperform RAND. RAND reaches an accuracy of 78% using 25 250 samples (ALI 99), ENT and BVSB reaches the same accuracy using 19 750 samples (ALI 77). We observe that the performance is highly identical during all ALIs for the active learners ENT and BVSB.

The selected samples from ALI 100 can be seen in Figure 5.8 and 5.9 for ENT and BVSB, respectively. Looking at these samples, we see that both learners are able to differentiate hard from easy samples.

We see a matching performance pattern as in EXP 1. Both active learners perform better than RAND; however, the difference in performance is small between ENT and BVSB.

As in EXP 1, we compare the class distribution of each learner. Since the learners select 250 samples during each ALI, it is easier to illustrate

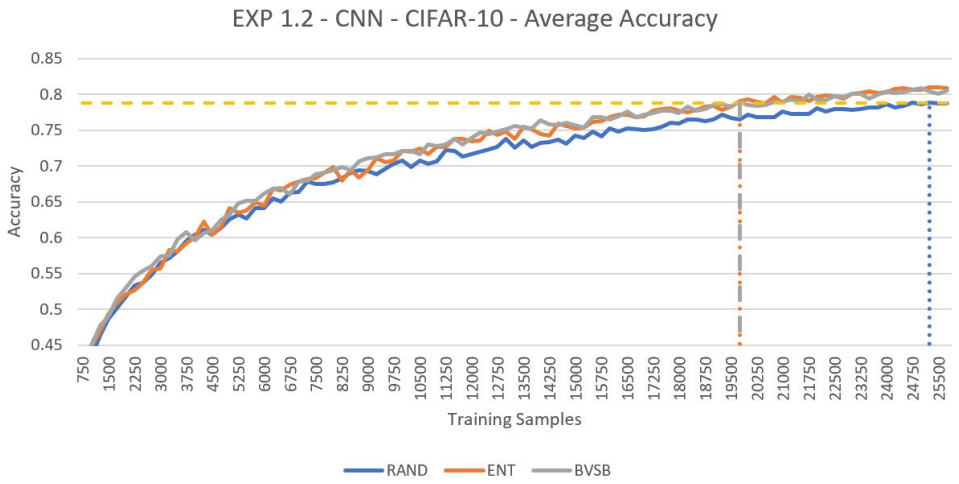



**Figure 5.6:** Performance results from EXP 1.2 by running AL with Image Classification on CIFAR-10. The accuracy is calculated using the test set of 10,000 samples, and averaged over three independent runs from ALI 1 to 50.

their preferred classes. Figure 5.7 presents how the class distribution of each learner changes over time from ALI 1 to 3, and what they end up with at ALI 100. We observe that BVSB is very balanced in selecting compared to ENT, which is more unbalanced and aggressive at collecting samples from specific classes. BVSB simulates a random selection similar to RAND by selecting highly informative samples, which makes it good at exploring (Joshi *et al.* [36]).

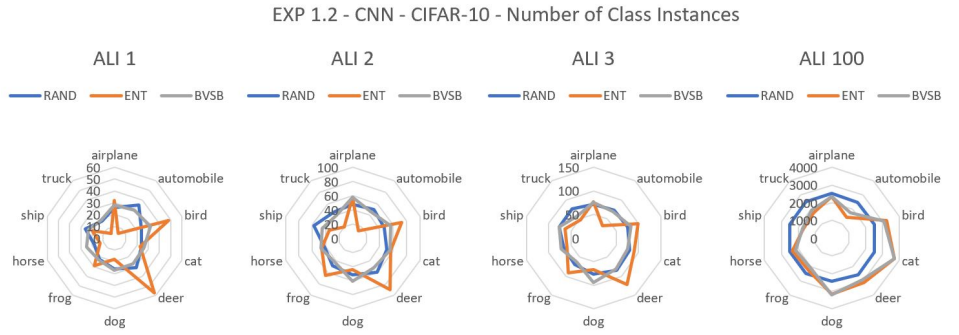
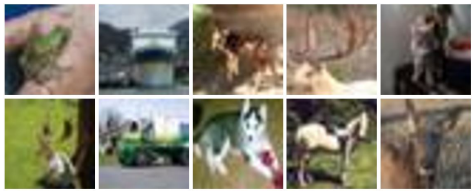



**Figure 5.7:** These plots illustrate how the number of class instances in the training sets change over time for each learner, from ALI 1, 2, 3, and 100.

**(a)** Classes: bird, cat, ship, bird, horse, cat, cat, truck, dog and airplane

**(b)** Classes: auto, frog, ship, horse, auto, ship, truck, ship, ship and ship

**Figure 5.8:** Images selected by ENT, from the CIFAR-10 dataset, at ALI 100. **Left:** Top 10 hard images with high informativeness scores. **Right:** Top 10 easy images with low informativeness scores.



**(a)** Classes: frog, ship, deer, deer, cat, deer, truck, dog, horse and deer

**(b)** Classes: bird, frog, auto, auto, horse, auto, frog, frog, auto and ship

**Figure 5.9:** Images selected by BVSB, from the CIFAR-10 dataset, at ALI 100. **Left:** Top 10 hard images with high informativeness scores. **Right:** Top 10 easy images with low informativeness scores.

## 5.2 EXP 2 - AL with Object Detection

**Using a more Complex Framework**
**Task:** Object Detection
**Model:** Detectron2's Faster R-CNN
**Dataset:** YYMNIST

### 5.2.1 Problem Description

The previous experiments focus on image classification, and our next step is to use AL with object detection. We base this on Detectron2's Faster R-CNN [82], due to its broad usage, state-of-the-art results, and well-documented open-source code. As mentioned in Section 2.2.3, there are mainly three aggregation techniques; we use the multi-detection entropy learners MAXENT, SUMENT, and AVGENT. We use entropy as this is a popular metric used in AL.

### 5.2.2 Dataset

Since we already got promising results in EXP 1 using MNIST, we choose an object detection version of the dataset called YYMNIST [85]. We generate a set of 10 000 images and split it into a training, test, and unlabeled set. The initial training set consists of 32 randomly selected images, the test set contains 1000 randomly selected images and has a fixed size throughout the whole experiment, and the remaining images virtually create an unlabeled set of 8968 images.

### 5.2.3 Setup

We implement and use our own AL Framework, as described in Section 4.1. We use Detectron2's COCO-pretrained Faster R-CNN model [82] to train on object detection. RAND selects $n$ samples randomly. MAXENT, SUMENT, and AVGENT select 2000 random samples from the unlabeled set, run inference over them, and return top $n$ highly informative samples.

#### ALIs

We follow Detectron2's training configuration on a toy dataset with minor changes and train the model for 300 iterations during the initial ALI. In all the following ALIs, each learner is trained for 100 iterations. We repeat this process for 50 ALIs and follow the AL process as in EXP 1. We initially train a model using the 32 randomly selected images. Then, we use the initial model as a starting point for the learners RAND, MAXENT, SUMENT, and AVGENT. Each learner adds a set of 16 images to the training set based on their informativeness score.

### 5.2.4 Results

A learner's performance can vary based on the initial set. We run the experiment three times using different initial training sets to get an average.

The performances can be seen in Figure 5.10, and Table 5.1 compares the performances reached by the active learners to the baseline RAND learner. The highest AP50 for MAXENT is 97.07 with 784 samples, for AVGENT, it is 96.98 using 816 samples, and for SUMENT, it is 97.45 using 816 samples.

During each ALI, MAXENT, SUMENT, and AVGENT select images having high and low informativeness. A comparison of these images can be seen in Figure 5.11, 5.12, and 5.13, respectively.

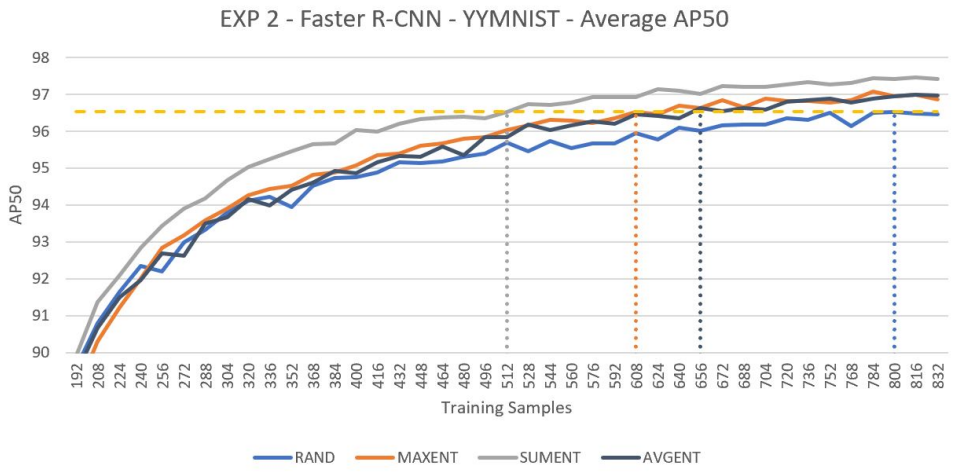



**Figure 5.10:** Performance results from EXP 2 using AL with Object Detection on YYMNIST. The performance is measured on a test set of 2000 samples, and averaged over three independent runs from ALI 10 to 50. Each run takes approx. 14 hours.

| | RAND | AVGENT | MAXENT | SUMENT |
|---|---|---|---|---|
| AP50 | 96.52 | 96.62 | 96.53 | 96.53 |
| Training Samples | 800 | 656 | 608 | 512 |
| ALI | 48 | 39 | 36 | 30 |

**Table 5.1:** Comparison of learners from EXP 2 using YYMNIST. This table presents the training samples and ALI of each active learner (dotted lines) when it reaches the same maximum performance as RAND (yellow dashed line) in Figure 5.10.

The results from this experiment presents the effects of using aggregation techniques when AL is applied on object detection. We notice that SUMENT tends to collect samples containing more instances compared to

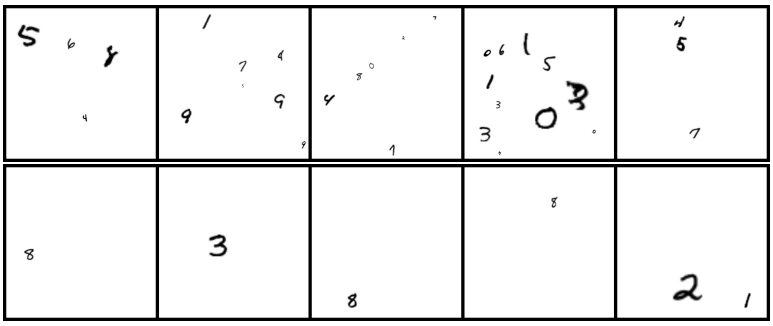


**Figure 5.11:** Top 5 hard images with high informativeness scores (top row) and top 5 easy images with low informativeness scores (bottom row); using MAXENT on the YYMNIST dataset.

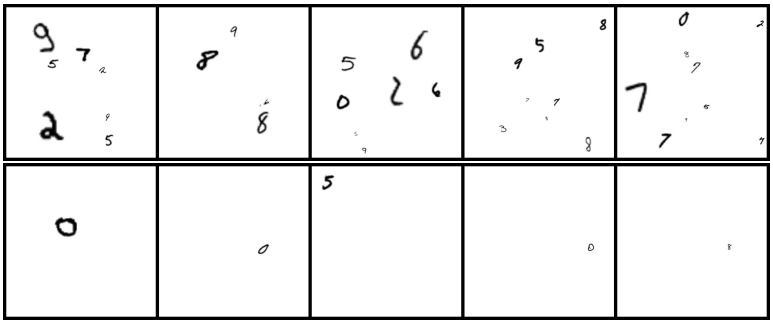


**Figure 5.12:** Top 5 hard images with high informativeness scores (top row) and top 5 easy images with low informativeness scores (bottom row); using SUMENT on the YYMNIST dataset.

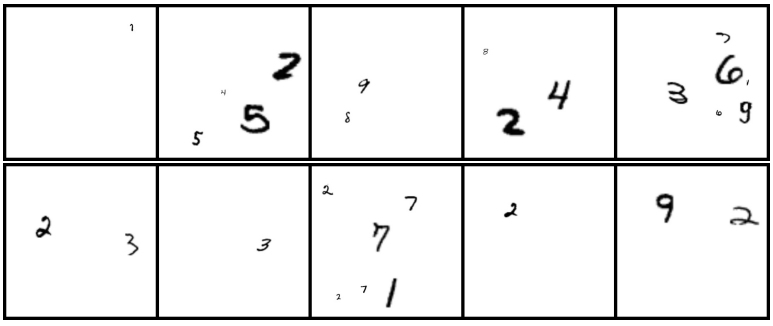


**Figure 5.13:** Top 5 hard images with high informativeness scores (top row) and top 5 easy images with low informativeness scores (bottom row); using AVGENT on the YYMNIST dataset.

the other learners. We confirm this by counting the instances in the final training set for each learner, and notice that SUMENT has a higher instance count. This can be seen in Figure 5.15a. In addition, we observe that AVGENT performs as MAXENT but better than RAND, even by having a lower instance count in the training set. Using an instance-rich training set does not necessarily lead to better performance. A lot of this behavior might be based on how the AVGENT learner scores images. Image A containing 10 detections can have the same average score as Image B containing a single detection. Figure 5.13 illustrates this behavior; AVGENT does not select images based only on high instance count as SUMENT.

### 5.2.5 EXP 2.1 - Using All Margin Learners

We rerun the experiment by introducing three new margin learners; maximum BvSB (MAXBVSB), sum BvSB (SUMBVSB), and average BvSB (AVGBVSB). We use the best-performing learner SUMENT for comparison.

In Figure 5.14, we observe that the margin learners perform better than the baseline RAND learner. However, there is not much difference in performance between the entropy and margin learners. More noticeably, the SUM learners (SUMENT and SUMBVSB) perform better than the AVG and MAX learners. Table 5.2 compares the performances reached by the active learners to the baseline RAND learner.

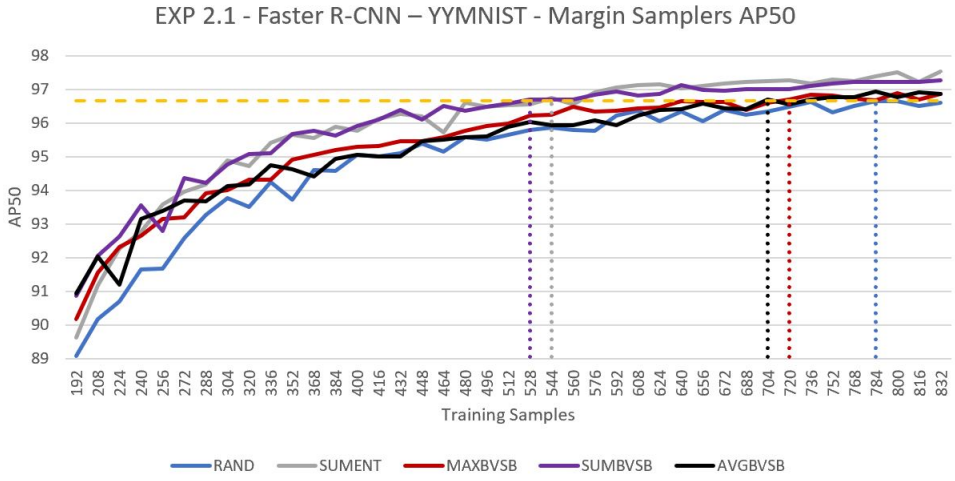


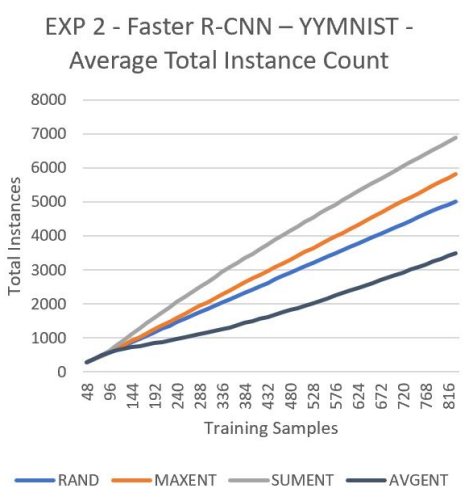

**Figure 5.14:** Performance results from EXP 2.1 using AL with Object Detection on YYMNIST. The performance is measured on a test set of 2000 samples, from a single run, and from ALI 10 to 50. Each run takes approx. 18 hours.

The margin learners MAXBVSB, SUMBVSB, and AVGBVSB, are not

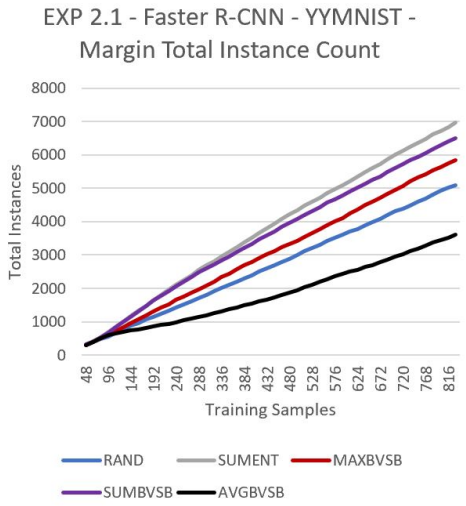
| | RAND | MAXBVSB | AVGBVSB | SUMENT | SUMBVSB |
|---|---|---|---|---|---|
| AP50 | 96.65 | 96.70 | 96.69 | 96.74 | 96.70 |
| Training Samples | 784 | 720 | 704 | 544 | 528 |
| ALI | 47 | 43 | 42 | 32 | 31 |

**Table 5.2:** Comparison of learners from EXP 2.1 using YYMNIST. This table presents the training samples and ALI of each active learner (dotted lines) when it reaches the same maximum performance as RAND (yellow dashed line) in Figure 5.14.

performing any better than the entropy learners as in EXP 1. We notice that a learner's performance is still highly affected by the aggregation techniques. As seen in Figure 5.15b, SUM still tends to select samples containing more instances than MAX and AVG.




**(a)** EXP 2 - Averaged total instances in training set each ALI from three runs




**(b)** EXP 2.1 - Total instances in training set each ALI from a singe run

**Figure 5.15:** These graphs show the number of class instances that are added to the training set during each ALI for each learner. Here, we see that the aggregation technique SUM favors images containing large number of instances, and that AVG selects images containing few instances.

## 5.3 EXP 3 - Synthetic Self-Driving Dataset

**Using a Synthetic Self-Driving Dataset**
**Task:** Object Detection
**Model:** Detectron2's Faster R-CNN
**Dataset:** Apollo Synthetic
**Extras:** Choosing samples from the entire informativeness spectrum

### 5.3.1 Problem Description

As part of our goal, we want to apply AL in a setting of autonomous driving for both object detection and instance-based segmentation. At the time of writing this thesis, the self-driving dataset Apollo Synthetic [74] is the right candidate that fits these tasks.

We gained much insight from EXP 2 in how each learner behaves using various aggregation techniques with object detection. To take our development a step further, we use the photo-realistic dataset with Faster R-CNN, and as in EXP 2, we use the learners MAXENT, SUMENT, and AVGENT for comparison.

### 5.3.2 Dataset

The Apollo Synthetic dataset has a well-organized folder structure, as described in Section 2.3.4. We extract RGB images, object ground truth files, and segmentation ground truth images from the dataset. After preprocessing, we end up with a random distribution of images shown in Table 5.3. Since this is an object detection task, we use all available thing classes in the Apollo Synthetic Dataset, as shown in Table 5.4.

| Sets | Size | Percent |
|---|---|---|
| Initial Training Set | 2 000 | 9.4% |
| Test Set | 3 000 | 14.1% |
| Unlabeled Set | 16 244 | 76.5% |
| Processed Dataset | 21 244 | 100.0% |

**Table 5.3:** Sizes of each set used for training and testing

| Thing Classes | | | | |
|---|---|---|---|---|
| Sedan | Hatchback | PickupTruck | Bus | Motorcyclist |
| SUV | Van | Truck | Cyclist | Pedestrian |

**Table 5.4:** The 10 thing classes that are used from the Apollo Dataset for EXP 3

Images are obtained from clear weather conditions, high road surface quality, and outdoor areas. Images containing no object ground truth are discarded. The dataset is structured hierarchically.

When working with real-life sensor systems, most of the collected sensor data is saved as a stream of synchronized data. We transform the hierarchical structure into a stream-based structure, thus ensuring simplicity and scalability to new incoming data (See Appendix C.2 for more details). The dataset is reduced by 30% after processing the images due to dataset issues, giving us 21 244 images from originally 30 240.

### 5.3.3 Setup

We continue using our AL Framework with Detectron2's Faster R-CNN model for object detection, and implement a custom DataLoader to convert the Apollo Synthetic dataset into the correct Detectron2-format. RAND selects $n$ samples randomly. For the active learners, we use a faster sampling selection method where MAXENT, SUMENT, and AVGENT select 5000 random samples from the unlabeled set, run inference over them, and return $n$ informative samples from the entire informativeness spectrum (i.e., Spectrum). Spectrum is explained in more detail below.

#### ALIs

We use the configuration from EXP 2, but with some slight changes. We train the model for 2500 iterations during the initial ALI. In all the following ALIs, each learner is trained for 500 iterations. We increase the number of iterations due to the complexity of the data.

The AL process is repeated for 50 ALIs. An initial model is trained with 2000 randomly selected images from the initial training set. The trained model is then used as a starting point for the learners. During each ALI, the model is evaluated using the test set, and 250 new images are added to the training set based on their informativeness score.

#### Added Functionality - Spectrum

As explained in Section 2.2.6, the exploration vs. exploitation trade-off is a known dilemma in AL [8]. In the previous experiments, we use a highly explorative sampling method. Samples having the highest informativeness are selected to be labeled by an oracle.

When working with pre-processed datasets such as MNIST and YYMNIST, there exists little to no noise that makes the objects easily distinguishable from the background. However, objects in photo-realistic samples can be hard to detect due to the complexity surrounding them. There is always a chance that explorative methods might select outliers, while exploitative

methods might select similar samples [66]. An active learner would prefer to both explore new data and exploit what it already knows.

We tackle this problem by introducing a slightly different sample selection method from what is proposed by Roy *et al.* [66]. We want to analyze how using the whole informativeness Spectrum vs. No-Spectrum impacts a learner's performance, as explained in Section 2.2.4. We refer to "Spectrum" as both exploring and exploiting by selecting samples with high, medium, and low informativeness scores. "No-Spectrum" is referred to as only exploring by selecting top $k$ samples having the highest informativeness. In this experiment, we use the Spectrum method. A comparison between using Spectrum and No-Spectrum is presented in EXP 3.1.

We sort the informativeness scores that are obtained after inference, from highest to lowest, and collect the top 150 samples with high informativeness, bottom 25 with low informativeness, and 75 randomly with medium informativeness. This creates a set of 250 samples that is added to the training set by each active learner. The learner gets the chance to explore with the highly informative samples, and exploit with the less informative samples. This distribution (e.g., 150 high, 75 mid, 25 low) is favoring exploration, since we believe it might be beneficial for a learner to explore as early as possible. Nevertheless, we do not focus on tweaking this distribution as this is out of our scope.

### 5.3.4 Results

We run the experiment three times with different initial training sets to get an average. Figure 5.16 presents the average AP50 for each learner. Keep in mind, the results from the ten first ALIs are not shown in this figure due to their instability, as the learners might not have gained substantial knowledge about the data yet.

Both active learners are able to perform better than RAND using fewer samples, and we notice a matching performance pattern as in EXP 2. Table 5.5 compares the active learners with the baseline RAND learner.

Figure 5.18a contains ten images that are scored by SUMENT as highly informative. Most contain clusters of overlapping instances, instances that are positioned in the distance, and instances that are either hidden behind other objects or placed in dark areas. Images with low informativeness are shown in Figure 5.18b. These images often contain large, clearly visible, and easily detectable instances. MAXENT selects images containing a single, highly informative instance regardless of the number of total instances (Figure 5.19). AVGENT selects images containing few instances, and is not affected by the number of instances. An image containing a single highly informative instance can be scored equally as an image containing multiple highly informative instances since it only considers the average (Figure 5.20.

The performance results indicate that having an instance-rich training

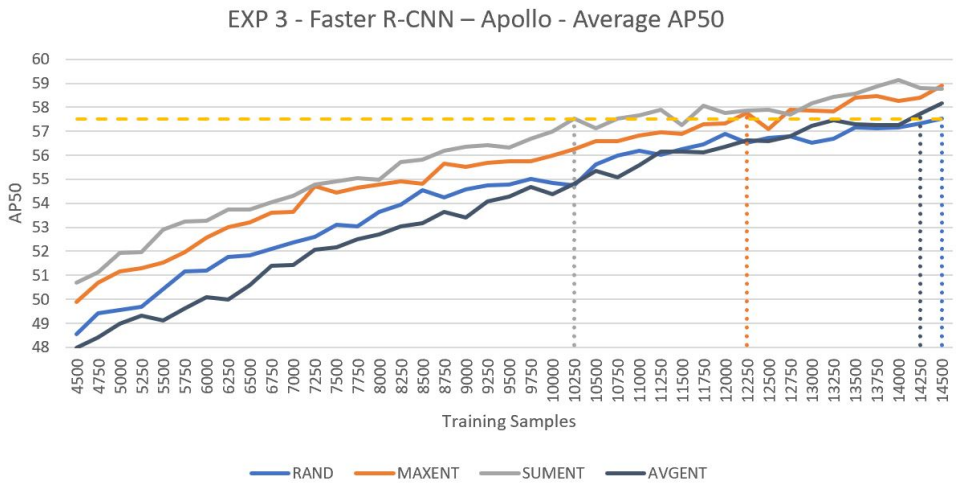



**Figure 5.16:** Performance results from EXP 3 using AL with Object Detection on Apollo Synthetic. The performance is measured on a test set of 3000 samples, and averaged over three independent runs from ALI 10 to 50. Each run takes approx. 2.5 days to complete.

| | RAND | AVGENT | MAXENT | SUMENT |
|---|---|---|---|---|
| AP50 | 57.52 | 57.71 | 57.74 | 57.52 |
| Training Samples | 14500 | 14250 | 12250 | 10250 |
| ALI | 50 | 49 | 41 | 33 |

**Table 5.5:** Comparison of learners from EXP 3 using Apollo. This table presents the training samples and ALI of each active learner (dotted lines) when it reaches the same maximum performance as RAND (yellow dashed line) in Figure 5.16.

set can be beneficial for an active learner, and that the aggregation technique SUM fulfills this requirement. The number of instances in the training set gives a performance boost and works in an active learner's favor in exploring new data. Figure 5.17 shows the training set instance count for each learner. Take notice that the instance count of the training sets for SUMENT, MAXENT, and AVGENT starts to meet at the end. Since our dataset contains 21 244 images, the training sets will start to match over time as more samples are added (i.e., the unlabeled set gets exhausted).

We further investigate why AVGENT performs poorly. In EXP 2, the YYMNIST dataset contains no "empty" images; there is always a digit. Apollo, however, can contain images having no true classes. As seen in Figure 5.20a, AVGENT tends to select images containing few or no true classes. If the average entropy score of an image containing multiple true positives is a bit lower than the average entropy score of an image containing

a single false positive, the latter is selected. AVGENT performs better than the RAND learner at the end of the ALIs since the unlabeled set is exhausted, as mentioned earlier; this can be seen in Figure 5.17.

In regards to our proposed selection method, a more direct comparison is needed to see if Spectrum makes the learners perform better than using No-Spectrum.

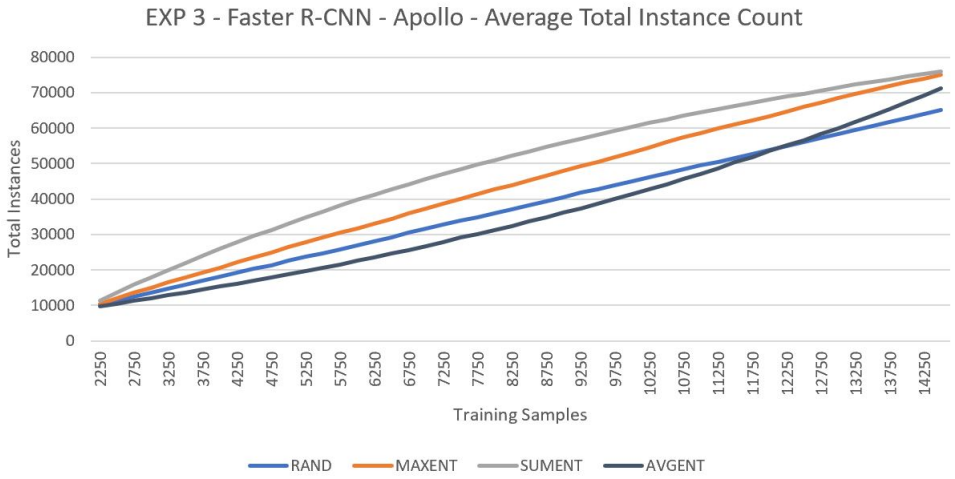



**Figure 5.17:** This graph shows the number of class instances that are added to the training set during each ALI for each learner. This illustrates that SUMENT favors images containing large number of instances. The values are averaged over three runs.


**(a)** Top 10 hard images with high informativeness using SUMENT

**(b)** Top 10 easy images with low informativeness using SUMENT

**Figure 5.18:** Scoring results on ALI 23 from using SUMENT on Apollo Synthetic.

**(a)** Top 10 hard images with high informativeness using MAXENT

**(b)** Top 10 easy images with low informativeness using MAXENT

**Figure 5.19:** Scoring results on ALI 23 from using MAXENT on Apollo Synthetic.

**(a)** Top 10 hard images with high informativeness using AVGENT

**(b)** Top 10 easy images with low informativeness using AVGENT

**Figure 5.20:** Scoring results on ALI 23 from using AVGENT on Apollo Synthetic.

### 5.3.5 EXP 3.1 - Spectrum vs No-Spectrum

We investigate the usage of the new sampling technique further by rerunning the experiment with three learners who use Spectrum and three learners who use the traditional selection method No-Spectrum. We compare the active learners MAXENT, SUMENT, and AVGENT. All learners use the same initially trained model as a starting point.

As seen in Figure 5.21, surprisingly, SUMENT benefits the most from using No-Spectrum, by only choosing highly informative images containing lots of instances. As we know from our earlier experiments, SUMENT gives images containing very few (sometimes one) easily detectable objects, a low informativeness score. Using Spectrum, images like these are selected from the low informativeness spectrum. Using No-Spectrum images are only selected from the high informativeness spectrum, and the learners end up with a training set containing images with lots of instances. MAXENT performs better using No-Spectrum, but the difference is not as big compared to SUMENT. As expected, AVGENT is not affected by this, since it does not take into consideration the number of instances in an image. In addition, AVGENT performs worse than RAND in the earlier ALIs; its performance improves when the training sets start to match at the end, as explained in EXP 3 results. Figure 5.22 illustrates this behavior by presenting the instance count of the training set for each learner using Spectrum and No-Spectrum.

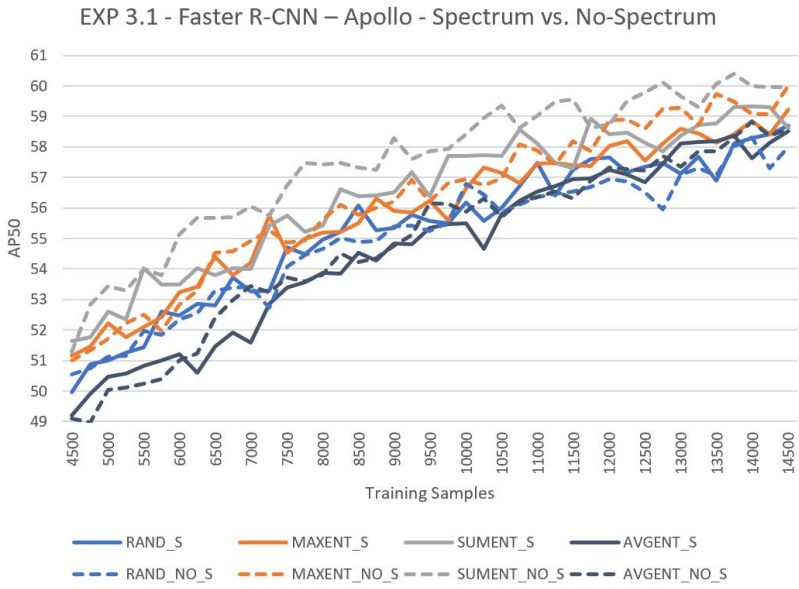



**Figure 5.21:** Performance results from EXP 3.1 using AL with the selection methods Spectrum (solid lines) and No-Spectrum (dashed lines). The performance is measured on a test set of 2000 samples. The values are of two runs from ALI 10 to 50. Each run takes approx. 3 days to complete.

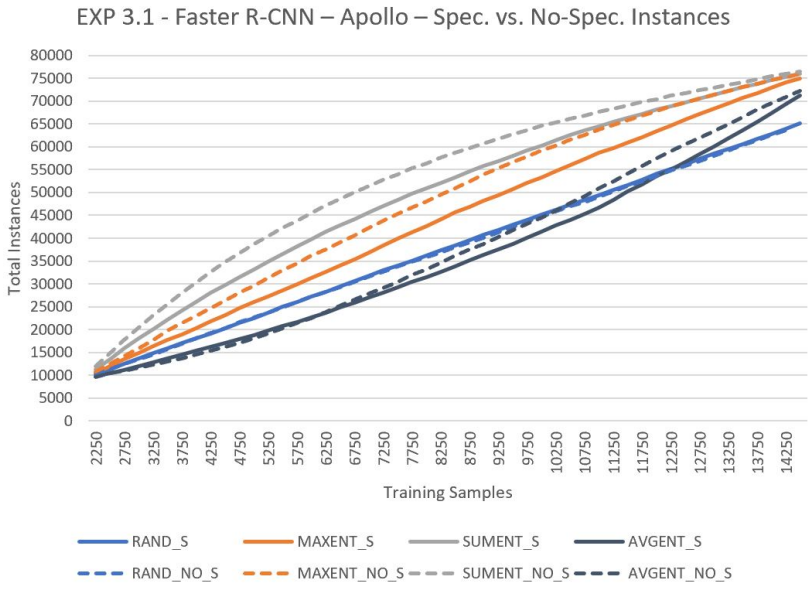



**Figure 5.22:** This graph shows the number of class instances that are added to the training set during each ALI for each learner using the selection methods Spectrum (solid lines) and No-Spectrum (dashed lines).

### 5.3.6 EXP 3.2 - Using an Unbalanced Initial Set

We perform a new experiment to examine how our learners behave when they are given an unbalanced initial model, which is trained using a highly unbalanced initial training set, as a starting point. The initial model is trained with 2000 images having the class distribution in Table 5.6. In this experiment, we use No-Spectrum and compare the entropy learners (e.g., SUMENT, MAXENT) with the margin learners (e.g., SUMBVSB, MAXBVSB).

| Unbalanced Initial Class Distribution | | | | |
|---|---|---|---|---|
| Sedan - 759 | Hatchback - 1 | PickupTruck - 0 | Cyclist - 0 | Bus - 0 |
| SUV - 1325 | Pedestrian - 1 | Motorcyclist - 0 | Truck - 0 | Van - 6 |

**Table 5.6:** We use an initial training set having the following unbalanced class distribution for EXP 3.2

As shown in Figure 5.23, SUMBVSB and MAXBVSB are able to perform slightly better than SUMENT and MAXENT in the early ALIs, respectively. We further analyze their performance by examining how their Average Precision for each class changes over time during these ALIs. Our focus is on rare classes such as Bus and Truck. The AP for each class is collected from ALI 0, 2, 4, 6, 8, 10 and 50, and the values are plotted for each learner in Figure 5.24. We notice that the margin learners detect these rare classes quicker than the other learners, since their APs for Bus and Truck increases early in the ALIs. As stated by Joshi *et al.* [36], BVSB is a fast explorer.

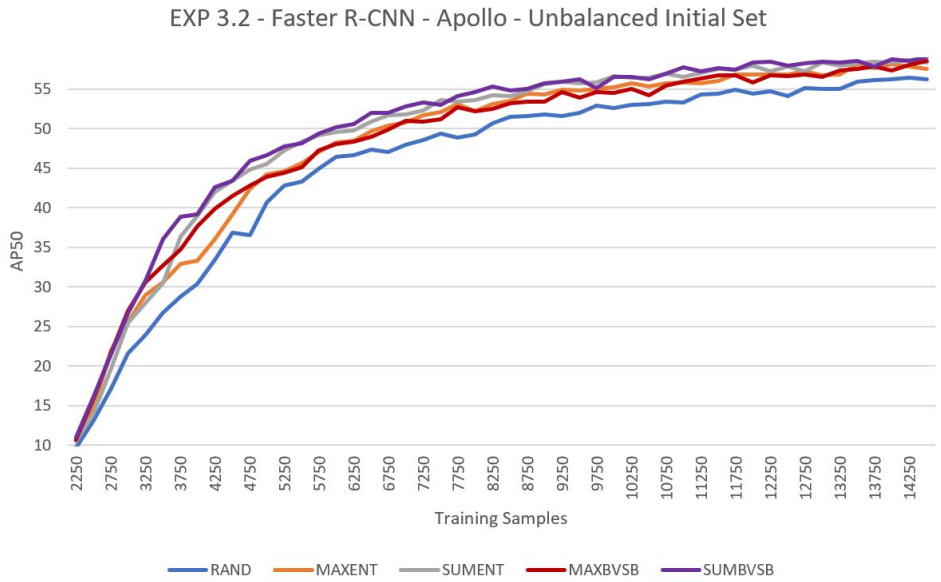



**Figure 5.23:** Results from EXP 3.2 using AL with an unbalanced initial model. The values are from a single run from ALI 1 to 50. The BVSB learners handles class imbalance better than the ENT learners early in the ALIs. This run takes approx. 3 days to complete.

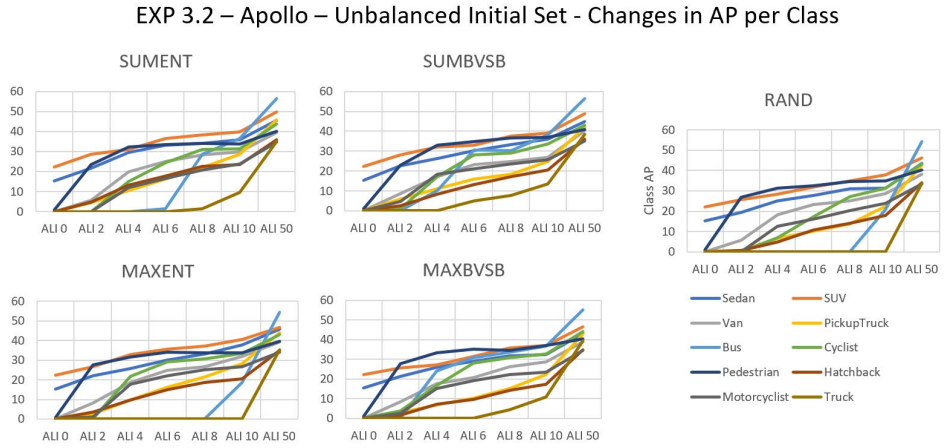



**Figure 5.24:** Expt 3.2 - Plots that illustrate how Average Precision (AP) increases over time, for each class, for multiple ALIs, and for each learner. SUMBVSB and MAXBVSB detect the classes Bus and Truck quicker than the other learners, which gives them a higher AP earlier in the ALIs.

## 5.4 EXP 4 - QbC Framework

**Using a more Complex Model**
**Task:** Object Detection and Instance-Based Segmentation
**Model:** Detectron2's Mask R-CNN
**Dataset:** Apollo Synthetic

### 5.4.1 Problem Description

Our next step is to apply AL on instance-based segmentation. Relevant AL approaches have been using Monte Carlo dropout to measure uncertainty in segmentation tasks (e.g., Sörsäter *et al.* [71]). In this experiment, we implement a QbC Framework using dropout layers following the approach proposed by Morrison *et al.* [54]. We examine if this type of framework can discover highly informative samples better than a simple Uncertainty Based framework. Measuring the informativeness of a sample using dropout layers appears to be a valuable approach that helps us understand a model's uncertainty and to improve its performance substantially. We use Detectron2's implementation of Mask R-CNN for this task, and the learners MAXENT, SUMENT, SUMBVSB, and DROPOUT for comparison.

### 5.4.2 Dataset

We continue using the Apollo Synthetic Dataset, as it contains ground truth data for instance-based segmentation. We use the same random set distribution and the pre-processed version of the dataset from EXP 3 having all thing classes. See Section 5.3.2 for details.

### 5.4.3 Setup

We implement and add the required functionality, such as a new DataLoader, a new training loop, and Dropout Layers, to our AL Framework so it can support and create a QbC Framework. We use Detectron2's COCO-pretrained Mask R-CNN Model for object detection and instance-based segmentation [82]. We use the two query strategy frameworks QbC and Uncertainty Based with the sample selection method No-Spectrum. No-Spectrum is used since it improves the performance of the learners, as seen in EXP 3.1.

For the QbC Framework, we use the learner DROPOUT. DROPOUT only selects 750 samples randomly from the unlabeled set, due to it being computationally heavy when it is running inference over a sample multiple times. This learner and the QbC Framework is explained in more detail in Section 5.4.3.

For the Uncertainty Based Framework, we use the learners MAXENT, SUMENT, and SUMBVSB. To make it fair and to make the results comparable with DROPOUT, the learners select 750 random samples each ALI

from the unlabeled set, instead of 5000 in earlier experiments. The learners then run inference over the samples, calculate their informativeness based on their predictions, and return the top $k$ samples having the highest informativeness score.

#### ALIs

We modify the configuration from EXP 3 by training the initial model in the initial ALI for 3000 iterations. In all the following ALIs, each learner is trained for 750 iterations. The number of iterations is increased due to the complexity of the model.

The AL process is repeated for 50 ALIs. We train the initial model with the initial training set having 2000 randomly selected images and use it as a starting point for the learners. During each ALI, each learner is evaluated on a test set, and 250 new images are added to the training set based on their informativeness score.

#### Added Functionality - Dropout Layers

Section 2.2.2 explains how a committee of learners can be used to find informative samples. Instead of using multiple learners, which can be computationally heavy, we implement a QbC Framework that creates this committee virtually by using dropout layers. These layers are added to the Fully Convolutional Network in Mask R-CNN, as proposed by Morrison *et al.* [54]. Using Monte Carlo dropout, we can run inference over the same set of images multiple times to measure the uncertainty.

We implement the learner DROPOUT following Morrison's *et al.* [54] approach. DROPOUT runs inference over the same set of images 5 times and measures informativeness based on the difference in predictions (e.g., semantic, spatial mask, and spatial bbox uncertainty). Its behavior has been explained in more detail under Section 4.2.1. During the earlier experiments, SUM had the best overall performance and is a good technique for finding instance-rich images; therefore, DROPOUT uses SUM as an aggregation technique.

### 5.4.4 Results

We run this experiment only once due to its complexity and the computationally heavy QbC Framework. Running once takes approximately 1 week. The AP50 of bounding boxes for each learner is shown in Figure 5.25, and for masks is shown in Figure 5.26. We see that all learners perform better than the baseline RAND learner, and SUMENT performs slightly better than MAXENT. However, the computationally heavy learner DROPOUT, did not perform any better than the uncertainty based learners. Figure 5.27 presents the total instances in the training set for each ALI for each learner.

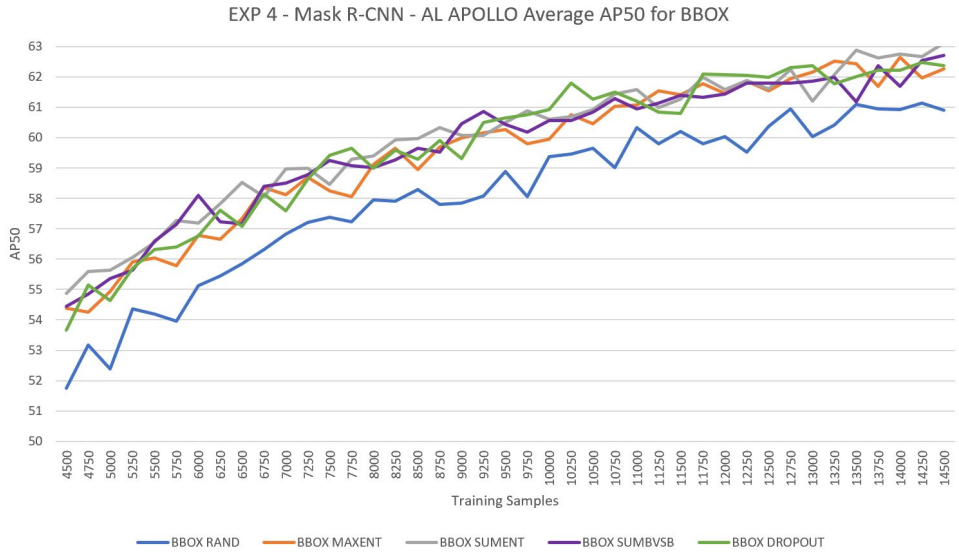



**Figure 5.25:** Results from EXP 4 using AL with Object Detection and Instance-Based Segmentation on Apollo Synthetic. The values are from a single run from ALI 10 to 50. AP50 is given for the bounding boxes. This run takes approx. 1 week to complete.

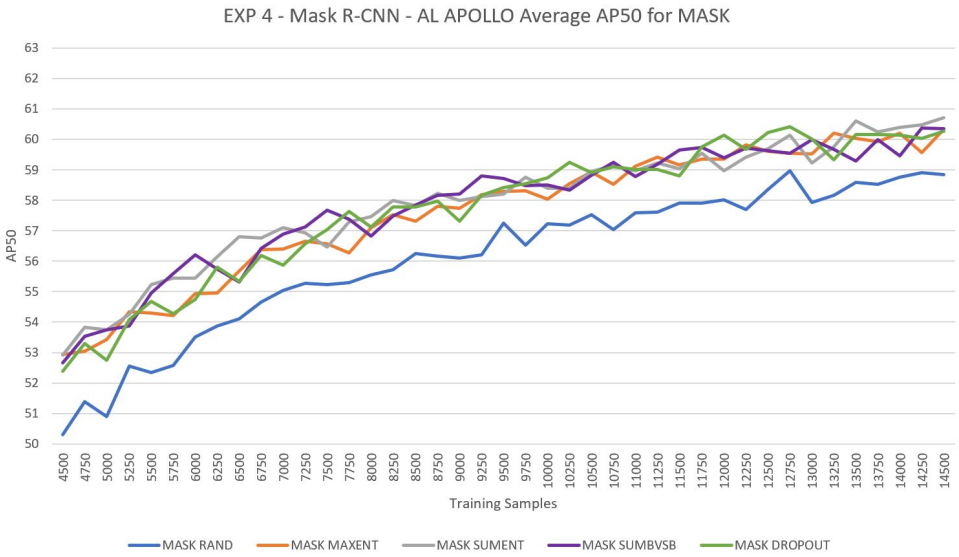



**Figure 5.26:** Results from EXP 4 using AL with Object Detection and Instance-Based Segmentation on Apollo Synthetic. The values are from a single run from ALI 10 to 50. AP50 is given for the masks. This run takes approx. 1 week to complete.

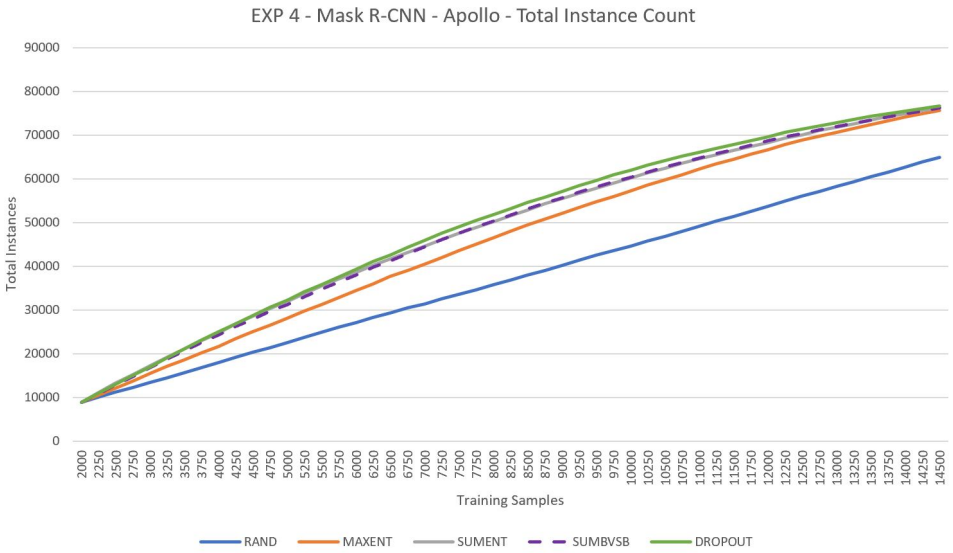



**Figure 5.27:** This graph shows the number of class instances that are added to the training set during each ALI for each learner in EXP 4. DROPOUT ends up with a overall more instance-rich training set.

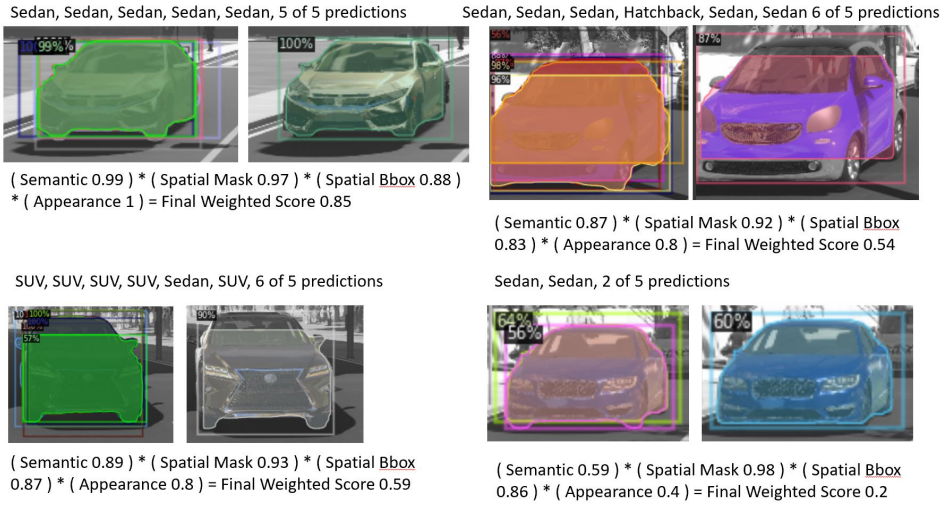



**Figure 5.28:** Scored observations from EXP 4 using AL with Object Detection and Instance-Based Segmentation on the Apollo Synthetic Dataset. Illustrates how uncertainty is measured using DROPOUT (Morrison *et al.* [54]). **Top Left:** A Sedan is given a high final weighted score, which results in low informativeness. **Top Right and Bottom Left:** A Sedan and a SUV are given final weighted scores based on the different uncertainty measures. **Bottom Right:** A Sedan is given a low final weighted score mostly due to few appearances, which results in high informativeness.

As explained in Section 3.3.1, each observation in an image contains a group of predictions that include softmax vectors, bounding boxes, and segmentation masks. Figure 5.28 presents four observations that are given an uncertainty score using DROPOUT and how their values are calculated. An observation's uncertainty is measured using four uncertainty measures based on the predictions. If we run inference over an image five times, we expect that the same instance is predicted an equal amount of times by the model (i.e., 5 out of 5 gives an appearance score of 1). We penalize if the appearances are too many or too few (i.e., 6 out of 5 or 4 out of 5). Semantic, spatial mask, and spatial bounding box uncertainties are calculated as well (See Appendix B for pseudo code).

Morrison *et al.* [54] performs a grid search across these uncertainty metrics to find good thresholds, as explained in Section 3.3.1 and 4.2.1. If the uncertainty measures are below these, the number of false positives increases, if above the number of true positives decreases. They state that it is necessary to find optimal thresholds, as doing so increases their overall performance.

We are able to successfully implement their method, and we observe that the DROPOUT learner is able to find informative samples as its performance is similar to the other active learners. However, we believe its performance can be further improved by tweaking and optimizing these thresholds on the various uncertainty measures. In addition, our results can not be directly compared with Morrison's *et al.* [54], since they do not provide any comparisons with a baseline. Currently, we prefer the simpler uncertainty-based learners due to them being less computationally heavier than DROPOUT.

## 5.5 EXP 5 - Real-Life Self-Driving Dataset

**Moving towards a Real-Life Dataset**
**Task:** Object Detection
**Model:** Detectron2's Faster R-CNN
**Dataset:** Waymo Open
**Extras:** Using the similarity metric LPIPS [86] to ensure data diversity.

### 5.5.1 Problem Description

In the preceding experiments, we apply AL using various toy datasets such as MNIST, CIFAR-10, and YYMNIST, as well as a synthetic self-driving dataset from Apollo. In this experiment, we explore how AL behaves in a realistic autonomous setting using a real-life dataset from Waymo. Detectron2's Faster R-CNN is used for this task, and the learners MAXENT, SUMENT, MAXBVSB, and SUMBVSB are used for comparison.

### 5.5.2 Dataset

We use Waymo Open v1.0 as our dataset, as explained in Section 2.3.5. We use a similar random set distribution as in EXP 3; initial training set of size 2000, test set of size 3000, and unlabeled set of size 24 154.

### 5.5.3 Setup

We use the AL Framework with Detectron2's Faster R-CNN, and implement a new custom DataLoader for the Waymo Open dataset. All learners use the No-Spectrum selection method by selecting 5000 random samples from the unlabeled set, run inference over them, and return top $n$ highly informative samples.

#### ALIs

We use the configuration from EXP 3. The initial model is trained for 2500 iterations during the initial ALI. In all the following ALIs, each learner is trained for 500 iterations.

The AL process is repeated for 50 ALIs. An initial model is trained with the initial training set having 2000 randomly selected images and is used as a starting point for the learners. During each ALI, we evaluate each learner on a test set, and 250 highly informative images are added to the training set.

### 5.5.4 Results

This experiment is run twice, the average AP50 of each learner is presented in Figure 5.29 illustrates the performances of each learner, and Table 5.7

compares the performances reached by the active learners to the baseline RAND learner. The results are highly different from what we observe in the earlier experiments. MAXENT has the best overall performance, the margin learners (e.g., SUMBVSB and MAXBVSB) perform slightly better than RAND early in the ALIs, but are able to increase their performance margin later at the end. AVGENT performs the worst out of all active learners, and we see a similar issue as discussed in EXP 3 Section 5.3.4; it is highly sensitive in collecting images containing false positives.

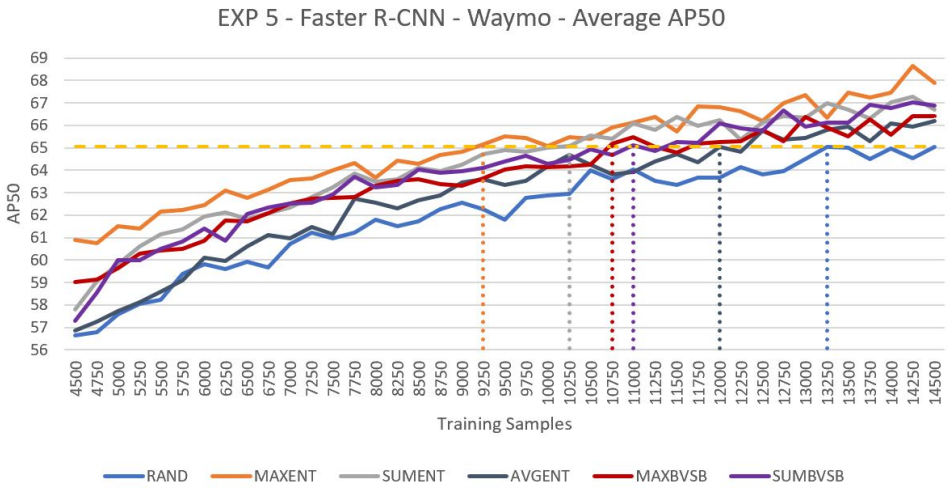



**Figure 5.29:** Results from EXP 5 using AL with Object Detection on Waymo Open. The values are from a single run from ALI 10 to 50.

| | RAND | AVGENT | SUMBVSB | MAXBVSB | SUMENT | MAXENT |
|---|---|---|---|---|---|---|
| AP50 | 65.05 | 65.02 | 65.12 | 65.17 | 65.08 | 65.13 |
| Training Samples | 13250 | 12000 | 11000 | 10750 | 10250 | 9250 |
| ALI | 45 | 40 | 36 | 35 | 33 | 29 |

**Table 5.7:** Comparison of learners from EXP 5 using Waymo. This table presents the training samples and ALI of each active learner (dotted lines) when it reaches the same maximum performance as RAND (yellow dashed line) in Figure 5.29.

By looking at the scored images from SUMENT, MAXENT, and AVGENT (Figure 5.31, 5.32, and 5.33, respectively), we notice that some are highly similar. We examine this further by looking into their set of informative samples.

Waymo Open contains video segments that are not pre-processed and diverse. If we select a frame from one of these segments, the surrounding frames will be very similar. The learners are affected by this. If a frame is given a high informativeness score, a surrounding frame will likely be given

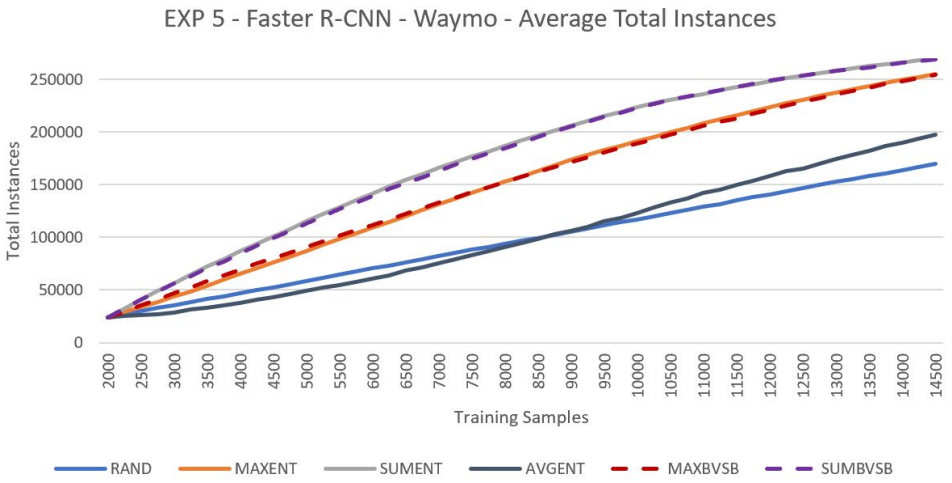



**Figure 5.30:** This graph shows the number of class instances that are added to the training set during each ALI for each learner in EXP 5.

the same score. In other words, there is a high chance that similar images are selected for labeling during each ALI; this does not lead to data diversity, as previously explained in Section 2.2.6.

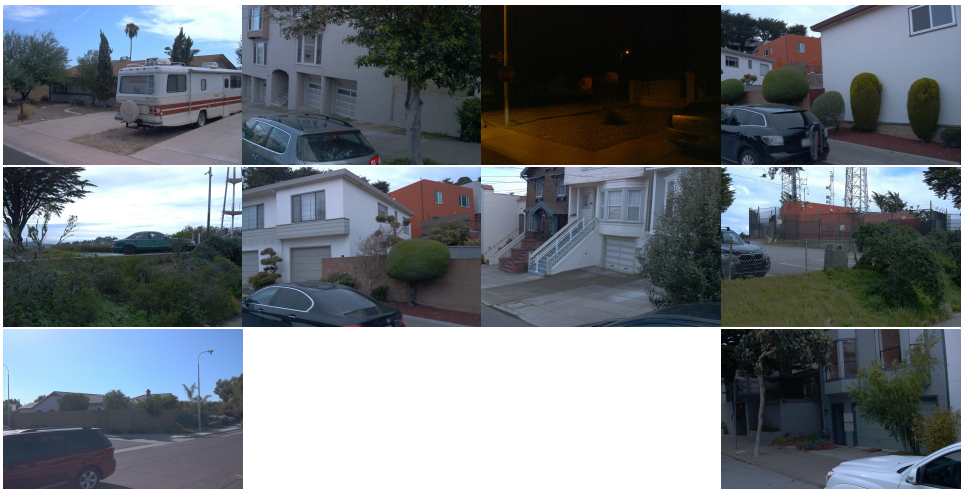
#### Added Functionality - LPIPS

We tackle this problem by adding new functionality, which ensures data diversity, to see if the performance can be improved. After inference, $n$ images are given an informativeness score and are sorted in decreasing order. We introduce an algorithm that uses the LPIPS metric (Zhang *et al.* [86]) for comparing images. We start by comparing the first highly informative image with $m$ subsequent images using LPIPS as metric. The reasoning behind comparing the $m$ closest images is to reduce computational cost. In addition, an image having high informativeness might not be similar to an image having low informativeness. Images having an LPIPS score lower than 0.4 are marked and ignored, as they are highly similar (e.g., 0.0 similar, 1.0 different). We repeat this process by comparing the next "unmarked" highly informative image with its following images. This is done until we have obtained a diverse set of 250 highly informative images. The similarity score of each image pair is saved for later reuse and is not computed multiple times as this is a time-consuming task.

### 5.5.5 EXP 5.1 - Ensuring Data Diversity

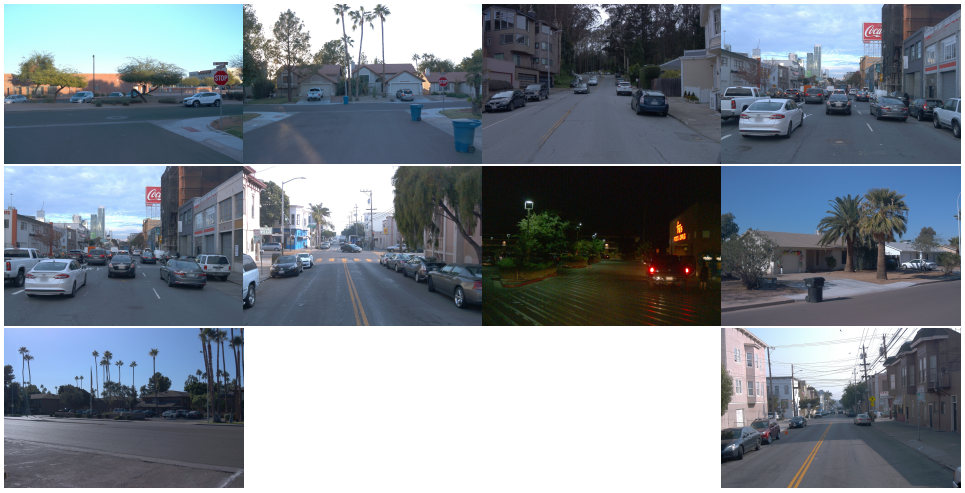
We rerun the experiment using the LPIPS metric (Zhang *et al.* [86]) to investigate if selecting a diverse set of highly informative samples lead to better performance. We follow the same setup as in EXP 5. By ensuring

**(a)** Top 10 hard images with high informativeness using SUMENT

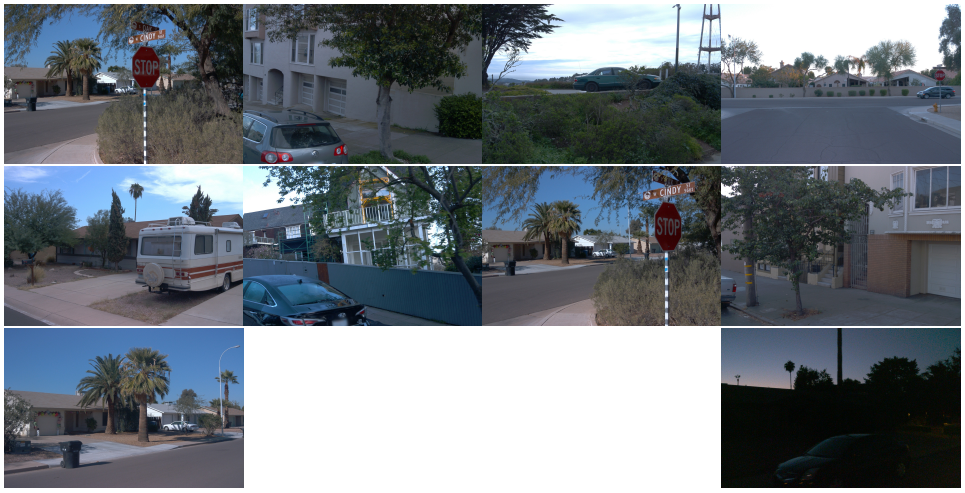


**(b)** Top 10 easy images with low informativeness using SUMENT

**Figure 5.31:** Scoring results on ALI 23 from using SUMENT on the Waymo Open dataset.

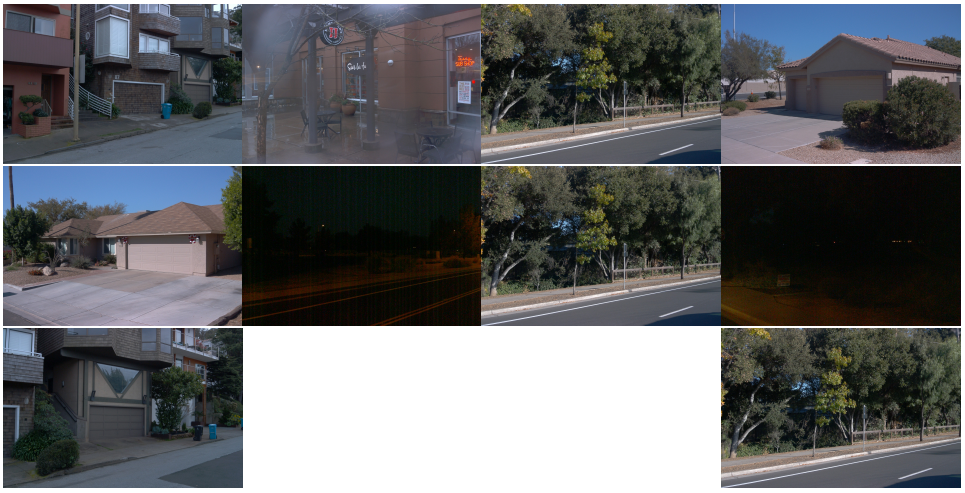


**(a)** Top 10 hard images with high informativeness using MAXENT



**(b)** Top 10 easy images with low informativeness using MAXENT

**Figure 5.32:** Scoring results on ALI 23 from using MAXENT on the Waymo Open dataset.



**(a)** Top 10 hard images with high informativeness using AVGENT

**(b)** Top 10 easy images with low informativeness using AVGENT

**Figure 5.33:** Scoring results on ALI 23 from using AVGENT on the Waymo Open dataset.

data diversity, we observe that the performance of MAXENT and MAXBVSB is drastically reduced early in the ALIs, SUMENT has a slight increase in performance, and the BVSB learners are not affected.

Based on our results, the MAX learners perform better by not using diverse samples. On this dataset, it might be that a model prefers multiple highly similar samples containing a single highly informative instance to become more certain; the model might want to exploit more than explore.

It is worth mentioning that the average runtime of the ENT learners is much longer compared to the BVSB learners; 3 days 13 hours and 1 day 10 hours, respectively. If our similarity algorithm receives a sorted set containing many similar images, it will take longer to create a diverse set. On the other hand, if the set contains few similar images, it will finish faster. In this experiment, the ENT learners fit the former case, while the BVSB learners the latter, since the ENT learners select more similar samples than the BVSB learners. Figure 5.35 illustrates the instance count in the training set each ALI for each learner. We observe that the total instance count of the ENT learners is reduced; this is more clear on SUMENT.

The algorithm removes similar samples with similar scores successfully from a list of informative samples to make it diverse. It is very time-consuming at first since we have to compare all image pairs; however, it gets faster over time since we save values, and it only compares image pairs from the same camera, since the dataset consists of images from three cameras. Nevertheless, a more robust technique is needed to create diverse datasets.

As a final note, by doing this experiment, we point out the weaknesses and challenges that may arise from using real-life video segments that contain multiple similar images, and how this can affect the performance.

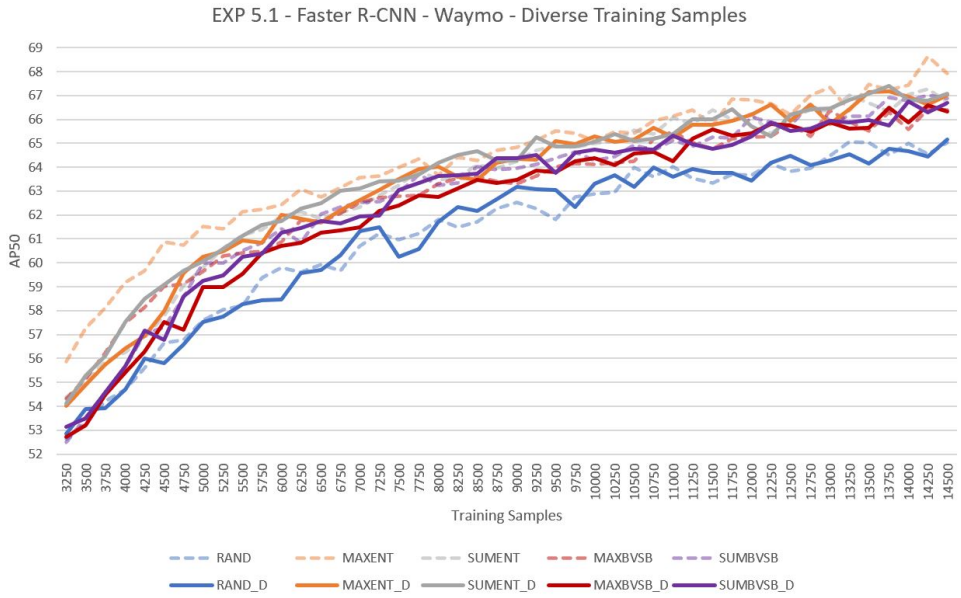



**Figure 5.34:** Results from EXP 5.1 using AL with Object Detection on the Waymo Open Dataset, with diverse training sets. The values are from a single run from ALI 5 to 50. This experiment takes approximately 9 days to run.

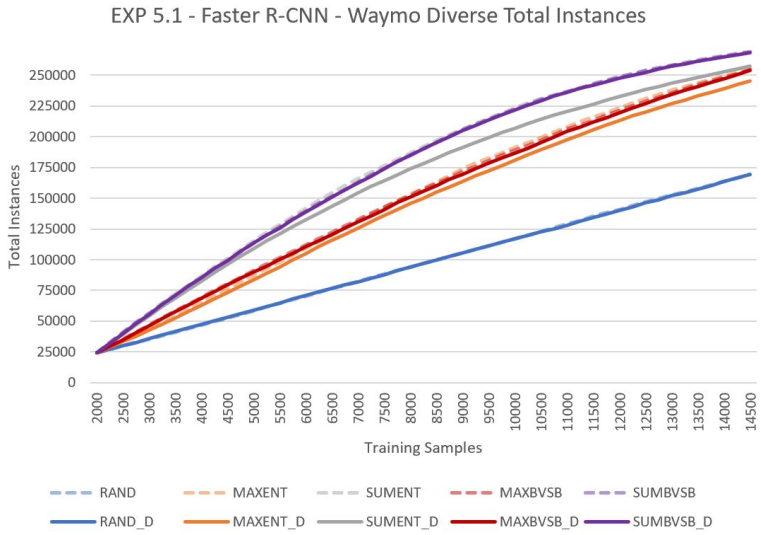



**Figure 5.35:** This graph shows the number of class instances that are added to the training set during each ALI for each learner. This is for EXP 5.1 that uses diverse training sets.

### 5.5.6 EXP 5.2 - Using Early Stopping

During our earlier experiments, we train each learner for a fixed number of iterations. This number is either set based on Detectron2's documentation for toy datasets, or manually tweaked after some initial test runs. Therefore, it is hard to tell if a learner is trained for too little or too long. We tackle this optimization problem by introducing early stopping in our AL Framework.

We rerun EXP 2, 3, and 4 using early stopping with No-Spectrum. The AL process is as follows. Initially, 2000 and 50 samples are randomly selected to create training and validation sets, respectively. The model is evaluated on the validation set every 100th iteration. For early stopping, we use 4 as patience and 0.005 as delta to monitor the difference in AP on the validation set. If the AP does not improve, early stopping is initiated. The initial model is then used as a starting point for the other learners. For each learner and during each ALI, highly informative samples are added to both the training and validation set. The training and evaluation processes are repeated using early stopping, and the experiments are run for 25 ALIs. We use the learners RAND, MAXENT, and SUMENT for comparison.

We are able to increase the performance using early stopping. However, at first, we observe a weakness by adding 250 highly informative samples each ALI to the training set, and 50 highly informative to the validation set, as mentioned earlier. Since highly informative samples are being added to the validation set, they are not used for training. The validation set will end up with collecting large amounts of informative samples for evaluation only, and these will never be used for training. After rerunning the experiment but by adding fewer highly informative samples to the validation set, the performance is increased as expected.

Figure 5.36, 5.37 and 5.38 shows a comparison of the performances for EXP 2, 3, 4, and 5.1 with and without early stopping, respectively. The results demonstrate that the learners have been training for too little, and by introducing early stopping their performances improve significantly on a smaller training set. Table 5.8 shows for how many iterations each learner is trained in total, with and without early stopping, for each run (e.g., YYMNIST, Apollo, Waymo), and from ALI 0 to 25. We believe that by tweaking the early stopping parameters, the performance can be further improved.

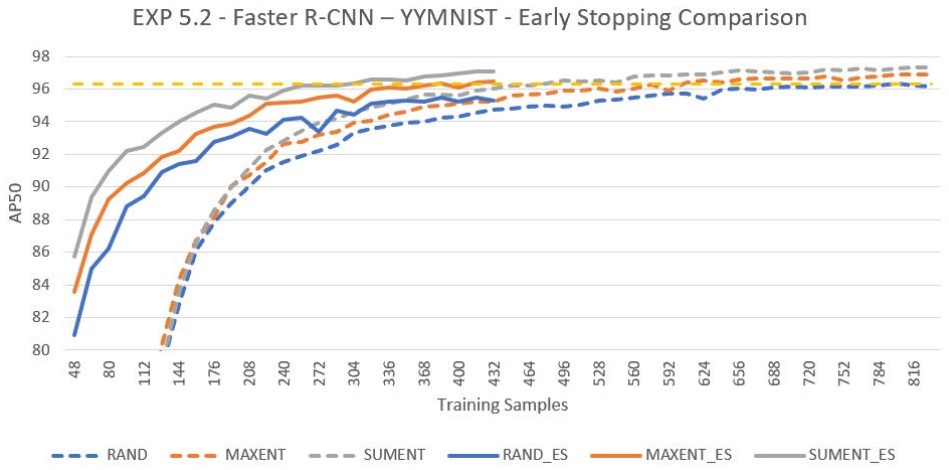



**Figure 5.36:** Results from EXP 5.2 using AL and Early Stopping with Object Detection on YYMNIST Dataset. The values are from a run with early stopping (ALI 1 to 25), which is compared to EXP 2 (ALI 1 to 50).

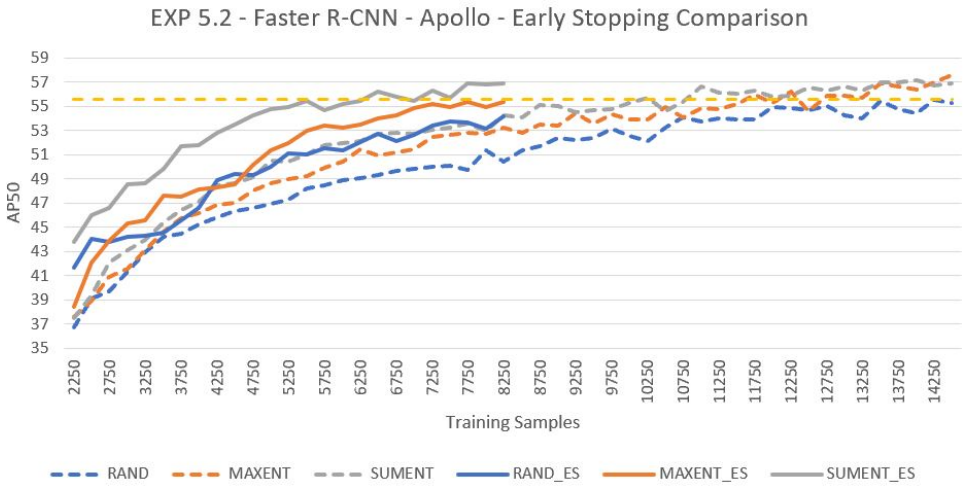



**Figure 5.37:** Results from EXP 5.2 using AL and Early Stopping with Object Detection on APOLLO Dataset. The values are from a run with early stopping (ALI 1 to 25), which is compared to EXP 3 (ALI 1 to 50).

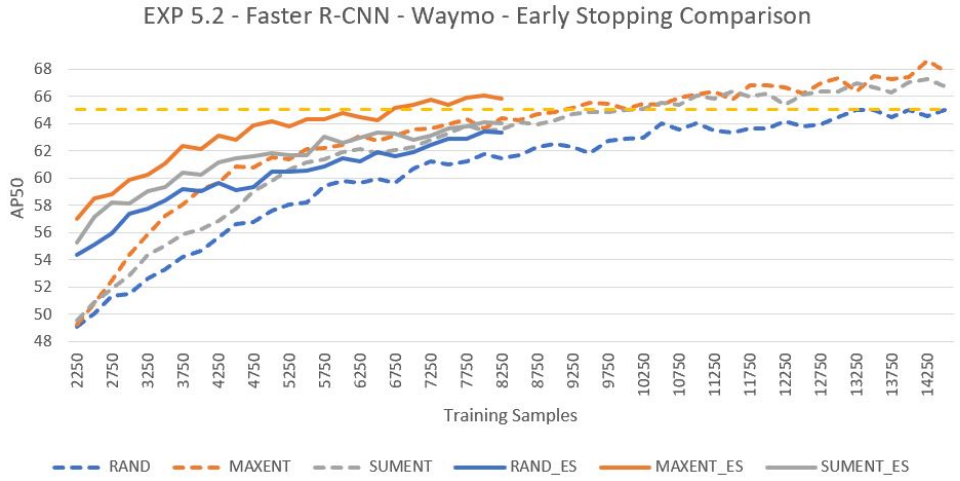



**Figure 5.38:** Results from EXP 5.2 using AL and Early Stopping with Object Detection on Waymo Open Dataset. The values are from a run with early stopping (ALI 1 to 25), which is compared to EXP 4 (ALI 1 to 50).

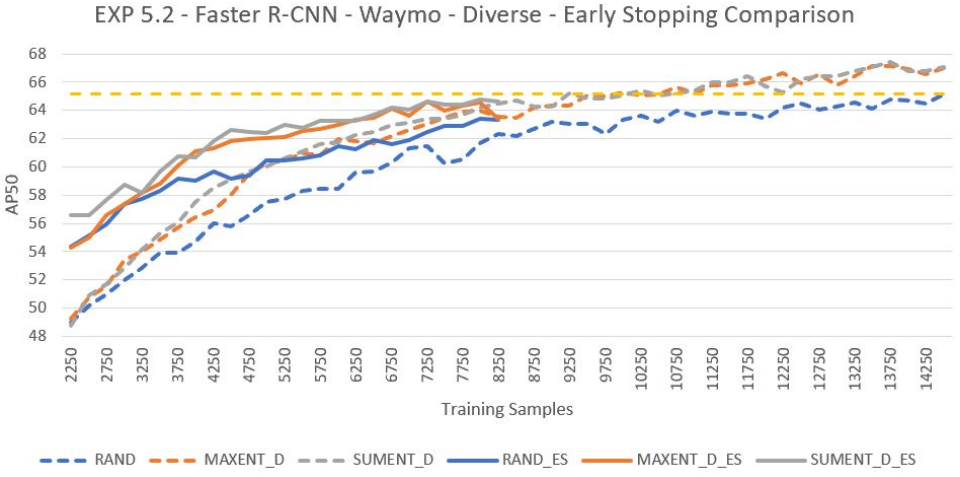



**Figure 5.39:** Results from EXP 5.2 using AL, Early Stopping and Data Diversity, with Object Detection on Waymo Open Dataset. The values are from a run with early stopping (ALI 1 to 25), which is compared to EXP 5.1 (ALI 1 to 50).

| Total Iterations | Init | RAND | MAXENT | SUMENT |
|---|---|---|---|---|
| YYMNIST - Without ES | 300 | 2500 | 2500 | 2500 |
| YYMNIST - With ES | 1800 | 17300 | 18200 | 18100 |
| Apollo - Without ES | 2500 | 12500 | 12500 | 12500 |
| Apollo - With ES | 2300 | 17500 | 17800 | 18100 |
| Waymo - Without ES | 2500 | 12500 | 12500 | 12500 |
| Waymo - With ES | 3800 | 28100 | 27600 | 24200 |

**Table 5.8:** This table shows the total iterations with and without Early Stopping (ES) for the initial set, and the learners. For each learner, iterations are summed from ALI 1 up to ALI 25. Here, we clearly see that the learners need more training, since their performances are improved using early stopping; however, it takes more time.

## 5.6 EXP 6 - Informative Samples

**Finding informative samples in NAPLab's raw dataset**
**Task:** Object Detection
**Model:** Detectron2's Faster R-CNN
**Dataset:** NAPLab Raw Data (NAP-Set)

### 5.6.1 Problem Description

As presented by Kale *et al.* [37], AL can be accelerated, by initializing it with transfer learning. Our goal is to find an initial set of informative samples from a never-before-seen dataset using transfer learning. For this experiment, we use two different models that are previously trained on self-driving datasets, to search for informative images on NAPLab's raw traffic data (NAP-Set). As the first model (Model-A), we use the best performing MAXENT model from ALI 50 in EXP 5, which is trained on Waymo Open using Faster R-CNN. As the second model (Model-B), we use one from Detectron2's model zoo that is trained on CityScapes using Mask R-CNN [82]. Keep in mind, Model-B is trained using the entire CityScapes dataset, while Model-A is trained using 50% of the Waymo Open dataset in EXP 5. Model-A uses the learners SUMENT, MAXENT, and AVGENT. Model-B uses all learners as Model-A, including DROPOUT.

### 5.6.2 Dataset

We use a small set of raw traffic data collected by NAPLab (NAP-Set).

### 5.6.3 Setup

All learners except DROPOUT select 5000 images from the NAP-Set, run inference over them, and measure their informativeness. DROPOUT selects

2000 images, runs inference over them 5 times, and measures their informativeness.

### 5.6.4 Results

Figure 5.41a and 5.41b shows hard and easy images selected by the two models using SUMENT. We observe that Model-B is able to select images that are more instance-rich and complex compared to Model-A. This is mostly due to Model-B being trained for longer on a full dataset, while Model-A being trained for less using AL in EXP 5. Other factors might be that both models are trained on different datasets. Model-A on Waymo Open, which contains images from the USA. Model-B on CityScapes, which contains images from Germany; Germany has a more similar city structure as Norway. We notice that Model-B, gives images containing high density and overlapping objects high informativeness, while images with low informativeness often contain few objects. Figure 5.41c and 5.41d shows hard and easy images selected by the two models using MAXENT, and Figure 5.42 by using AVGENT.

Figure 5.40 shows hard and easy images selected by Model-B using DROPOUT. We observe that Model-B using the DROPOUT learner collects even more instance-rich images compared to it using SUMENT. In DROPOUT, the uncertainty of predictions is measured using both the bounding box and the masks.

The important point to make here, is that transfer learning can be beneficial to obtain an informative initial set, presuming we have a good model trained on the same data domain, and to accelerate AL (Kale *et al.* [37]). Samples with high informativeness can be sent to a human for manual annotation, while samples with low informativeness can be used for assisted annotation to speed up the annotation process.

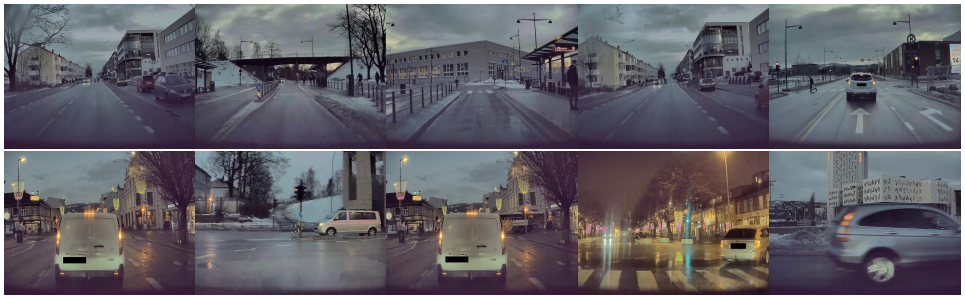
**(a)** Top 10 hard images - Model-B using DROPOUT

**(b)** Top 10 easy images - Model-B using DROPOUT

**Figure 5.40:** Images selected from the NAP-Set by only using Model-B: Fully trained CityScapes model from Detectron2 (Mask R-CNN). **5.40a:** Top 10 hard images starting with the hardest. **5.40b:** Top 10 easy images starting with easiest.

**(a)** Top 5 hard/easy images - Model-A using SUMENT

**(b)** Top 5 hard/easy images - Model-B using SUMENT



**(c)** Top 5 hard/easy images - Model-A using MAXENT

**(d)** Top 5 hard/easy images - Model-B using MAXENT

**Figure 5.41:** Images selected from the NAP-Set. Model-A: Best performing MAXENT learner from ALI 50 in EXP 5 (Faster R-CNN). Model-B: Fully trained CityScapes model from Detectron2 (Mask R-CNN). **Top row in each subfigure:** Top 5 hard images starting with the hardest. **Bottom row in each subfigure:** Top 5 easy images starting with easiest.

**(a)** Top 5 hard/easy images - Model-A using AVGENT

**(b)** Top 5 hard/easy images - Model-B using AVGENT

**Figure 5.42:** Images selected from the NAP-Set. Model-A: Best performing MAXENT learner from ALI 50 in EXP 5 (Faster R-CNN). Model-B: Fully trained CityScapes model from Detectron2 (Mask R-CNN). **Top row in each subfigure:** Top 5 hard images starting with the hardest. **Bottom row in each subfigure:** Top 5 easy images starting with easiest.

# Chapter 6

# Discussion

In this chapter, we discuss our major findings gained using the different Active Learning (AL) strategies. Furthermore, we shortly discuss important factors and limitations that need to be considered when applying AL. Finally, we address the Research Questions (RQs) listed in Section 1.3, followed by the limitations of our thesis.

Our goal is to identify if AL can be used to achieve high performance using a smaller, highly informative training set. By iteratively conducting experiments with increasing complexity, and in parallel implementing an AL Framework, we make interesting discoveries on how various AL strategies behave and perform.

## 6.1 Major Findings

Our results show that the active learners can perform better than the baseline RAND learner using less labeled training data. However, we observe that the number of total class instances in the training set has a great impact on a learner's performance. This is where the aggregation technique SUM stands out, as it is known to select images containing many highly informative instances. The aggregation technique AVG, behaves differently, it selects images containing fewer instances. There are, nevertheless, drawbacks with these types of aggregation techniques as they are highly sensitive to false positives.

### False Positives

The number of false positives predicted by a learner can be profoundly affected by the complexity of the dataset. In our experiments, we are using a toy, a synthetic, and a real-life dataset (e.g., YYMNIST, Apollo Synthetic, Waymo Open, respectively).

By picking an arbitrary image from YYMNIST, we see that there are always non-overlapping digits placed randomly on top of a white background.

This minimizes the chance of predicting false positives. For an image taken from Apollo, the complexity increases drastically, the images are no longer as consistent, and can be highly diverse.

For an image taken from a real-life dataset, the complexity is even higher, as there might contain high amounts of noise and unimportant data, and it might even contain classes of no interest or no instances at all. In such circumstances, the number of false positives can increase considerably.

As our results present, we see a similar performance trend in the pre-processed toy and synthetic datasets. However, applying AL on a real-life dataset gives different results. SUMENT has the best performance in most experiments, but the negative impact of including false negatives can be seen in EXP 5, where it performs poorly compared to the other learners. AVGENT has the worst overall performance as it tends to score images containing a single, low confidence, false positive, as highly informative. In addition, most of the images selected by AVGENT contain few true positives, as discussed in Section 5.3.4. This weakness can be improved by increasing the prediction confidence threshold used under testing to discard low confidence detections that might be false positives (e.g., we are using 0.5, and Detectron2 uses 0.7 as default).

The aggregation techniques sum, maximum, and average are presented by Brust *et al.* [9]. In their approach, sum is best-performing, while maximum and average achieve similar performance. Our learners SUMENT and MAXENT reflect their results; however, AVGENT does not. This is due to the usage of different datasets. They use PASCAL VOC, where all images contain at least one object, while we use datasets containing images having no objects.

#### Instance-Rich Samples

We notice that the object detection learners end up with varying numbers of total instances in their training sets. One interesting fact, in EXP 2 (Section 5.2.4), AVG still manages to perform better than RAND, and gets a similar performance as MAX (similar to what is found by Brust *et al.* [9]), while having the lowest total instance count. Here, we use the dataset YYMNIST, which contains samples having at least one object. The SUM learners, having the highest instance-count, benefit from selecting images containing many objects, as most of these might be highly informative. In addition, these might contain hard-to-detect and rare objects that result in diverse scenarios in the context of autonomous perception. Brust *et al.* [9] use these aggregation techniques with margin learners (e.g., 1-vs-2), and observe a similar behaviour in their approach (Section 3.2.3).

### Exploration vs. Exploitation

Selecting a set of samples containing many informative instances can be seen as an explorative method for the learners (i.e., using SUM). However, as we explained in Section 2.2.6, there are trade-offs between exploring and exploiting. We introduce two techniques to evaluate this in two different experiments. Keep in mind that these results can not be compared directly due to the usage of different datasets (e.g., synthetic, real-life).

In EXP 3.1 (Section 5.3.5), we propose the selection methods Spectrum and No-Spectrum. As we evaluate each active learner using a different aggregation technique, we discover that SUM does not benefit from using Spectrum, and that AVG and MAX learners are not affected.

In EXP 5.1 (Section 5.5.5), we propose a technique to select diverse samples that are highly informative. Diversifying training samples can be seen as a way of exploring. Using a real-life dataset containing video segments (e.g., Waymo Open) gives multiple frames that are highly similar. If a frame is scored as highly informative, there is a chance that the surrounding frames will be given a similar score. If so, the learner will select similar images, and become somewhat exploitative, as no new information is gained. The performance of MAX learners is reduced drastically by using diverse training samples. It might be that these learners want to exploit more when using this particular dataset than explore.

The explore vs. exploit trade-off has to be investigated further, as our proposed methods result in different performances depending on the dataset (e.g., synthetic, real-life). Nevertheless, our results demonstrate that highly explorative learners achieve better performance by selecting instance-rich samples early in the AL process; hence, SUM learners using No-Spectrum.

### Instance-Based Segmentation - Strategy Complexity

Complex AL strategies can be hard to tune, computationally expensive, and might not lead to high performing learners compared to simpler AL strategies. To our knowledge, few strategies are proposed to measure uncertainty on instance-based segmentation (e.g., Sörsäter *et al.* [71] and Morrison's *et al.* [54]), thereby making it difficult to compare the results. We follow Morrison's *et al.* [54] approach by introducing an additional uncertainty measure and implement a QbC Framework. Even though the results are not as promising, it is worth discussing the complexity of this framework; it has to run inference on the same set of images multiple times. When creating an AL strategy, there is a trade-off between the complexity and time-consumption, and its ability to output a high performing learner. In other words, if an AL strategy is computationally expensive, and gives a worse performing learner compared to a computationally light strategy, we believe, it may not be worth using. Nevertheless, as mentioned in EXP 4

(Section 5.4.4), their approach needs to be further optimized.

#### Early Stopping

Using early stopping with AL gives a huge performance boost, as demonstrated in EXP 5.2 (Section 5.5.6). We do not focus on optimizing early stopping; however, there is still a question left. How should samples be added to the validation set each ALI? In EXP 5.2, we only add a small set of highly informative samples to the validation set. Other ways can be to add a random set of samples, or by adding a set containing samples with both low, medium, and high informativeness. An optimal case would perhaps be to have a validation set that represents the underlying data structure in the best possible way (e.g., being class balanced).

## 6.2 Softmax Uncertainty

Our work is based on measuring uncertainty using softmax vectors, as this is the output we obtain from the models (e.g., Faster R-CNN and Mask R-CNN) in Detectron2. However, using the probability distribution over all classes might not be a good uncertainty measure.

Gast *et al.* [23] explores how uncertainty can be measured in neural networks in the context of probabilistic deep networks. They further discuss the weakness of a softmax vector, as it only predicts which class is highly certain in relation to other classes and not the uncertainty of the network itself. In addition, they mention that softmax uncertainty is not well-calibrated and that this drawback is known in deep learning papers [5, 20, 28]. An image containing an unknown class can still be predicted as a known class with high probability (Bendale *et al.* [5]). A model can predict a class with high probability having a high softmax output, but still be highly uncertain (Gal *et al.* [20]).

## 6.3 Bounding Box vs. Mask Uncertainty

In most experiments, we measure the uncertainty based on softmax vectors only. In EXP 4, we use dropout layers to run inference over images multiple times, and to measure the uncertainty on predictions, we end up with a set of multiple bounding boxes and masks for each prediction. As explained in Section 3.3.1 and 4.2.1, overlapping predictions are grouped into observations, and later used to measure spatial bounding box and spatial mask uncertainty. Morrison *et al.* [54] does not use bounding boxes to measure uncertainty. They state that a set of masks is advantageous when combining detections compared to bounding boxes, as the set of masks much likely will

represent the same object, in scenarios where we are introduced to tightly grouped or irregularly shaped objects. We illustrate this in Figure 6.1.

(a) (b) (c)

**Figure 6.1:** We run inference over a single image 10 times from Apollo Synthetic, using the trained DROPOUT learner from ALI 50 in EXP 4. A set of overlapping masks are more likely to represent an object compared to a set of bounding boxes (Morrison *et al.* [54]). **Figure 6.1a:** The ground truth of this image. **Figure 6.1b:** All predicted bounding boxes after 10 inferences put on the same image. **Figure 6.1c:** All predicted masks after 10 inferences put on the same image.

Since the domain of self-driving datasets is highly complex, we are frequently introduced to images containing multiple overlapping objects. Bounding boxes do not give as much information about the objects as masks or segments, which contain vasts amounts of information used to obtain good scene understanding [42]. Whether uncertainty should be measured using only bounding boxes (Kao *et al.* [39]), only masks (Morrison *et al.* [54]), or both has to be further investigated.

## 6.4 Initial Training Set

The final performance of a learner is heavily affected by the samples selected in the initial training set. In our image classification experiments, we use a balanced initial set containing an equal amount of samples from each class. In the later experiments, we follow typical AL (i.e., initially selecting samples randomly from the unlabeled set) to create an initial training set. We believe that creating a good initial training set is essential to take advantage of AL fully.

In an AL setting, limited time and annotation resources can affect the initial training set's size. For instance, an easy approach to create this set, is to select 2000 samples randomly from the unlabeled set and annotate, which is what we do. A more favorable approach would be to extract a set of 1000 samples that is diverse, instance-rich, and has little class imbalances. Kang *et al.* [38] reaches high accuracy using AL in text classification, by clustering samples and selecting those being closest to the centroids as the initial training set. Nonetheless, making this initial set can become a challenging

task in an autonomous real-life setting related to object detection. The image similarity metric LPIPS (Zhang *et al.* [86]) might be a good candidate for selecting similar images in the context of traffic scenarios.

## 6.5 Time and Annotation Cost

Since data is often manually annotated by an expert, it is considered as a bottleneck in the annotation process. Raw sensor data can be easy to obtain, but costly and time-consuming to process and annotate.

AL reduces the annotation job, since a human annotator ends up annotating a smaller set of informative samples, instead of annotating the entire available dataset. However, the cost of annotating an informative sample itself can vary. For instance, we notice that learners using SUM tend to select images containing multiple instances. For learners using MAX, this number can be less. Annotating informative images selected by SUM might take more time than an image selected by MAX.

Settles *et al.* [69] presents an empirical study that investigates the annotation cost of AL in a real setting. They conclude that the annotation cost is not constant for all instances in an image and that it can vary depending on the human annotator. Furthermore, they state that the annotation time can increase if the informative samples are difficult, and they argue that the annotation cost has to be incorporated in an AL process to truly reduce the labeling cost. As presented in Section 3.3, the approach proposed by Mackowiak *et al.* [51] is such an example. They minimize the manual annotation effort even further in semantic segmentation by giving the annotator highly informative regions in an image to annotate.

## 6.6 Research Questions (RQs)

**RQ1:** Can we use AL to achieve better or similar performance with less labeled data as opposed to utilizing the entire dataset? If so, how much time and resources can be saved in collecting, annotating data, and training in the setting of autonomous perception?

Our results demonstrate that most active learners can achieve a better overall performance than the baseline learner. In addition, most active learners can reach the same maximum performance as the baseline learner using a smaller training set.

When it comes to time and resource savings, AL primarily affects the annotation process by minimizing the time needed to annotate. In this thesis, we do not manually annotate samples, as the data is already labeled. Therefore, we do not get the chance to monitor the manual annotation process itself. Still, our experiments indicate that a small amount of annotated

training data can achieve satisfactory performance, thereby save time used for annotation. For instance, in an autonomous perception setting, we have a large amount of sequential data coming in from multiple cameras. Some sub-segments contains highly similar images, (e.g., when the vehicle is standing on a red light or is parked), or images with little information (e.g., when camera sensors are blocked by other vehicles or objects). In these scenarios, a human annotator can focus on manually annotating highly informative images given by the learner. Furthermore, images given low informativeness often contain confident detections. These detections can be used for assisted annotation in parallel to manual annotation and thereby speed up the annotation process.

Even though the annotation time is minimized by only annotating a small informative set, it does not necessarily mean that the annotation time on the samples themselves is reduced, as explained in Section 6.5. In EXP 6 (Section 5.6.4), we simulate a scenario to obtain informative samples from a never-before-seen dataset using transfer learning, and observe that the highly informative samples selected by DROPOUT and SUMENT can become difficult to label, as they are instance-rich, compared to MAXENT and AVGENT. Nevertheless, we explain that an advantage would be to use predictions from the model for assisted labeling.

In traditional supervised training, a model is often trained once on a labeled dataset, while in AL, the model is repeatedly trained with an increasing training set. In some scenarios, AL can make a model reach high accuracy faster than traditional learning. In addition, based on our results, we see that the active learners can achieve the same performance as the baseline learner much quicker. There is a trade-off between costly human labor (e.g., manual annotating) and cheap processing power (e.g., model training) that has to be considered. For instance, traditionally, humans will use a lot of time to annotate data, then train a model on the entire dataset. With AL, this is done in a loop. Humans will use little time in annotating a small set of data, then train the model without using the entire dataset. Here, we see that the overall training time (i.e., processing power) using AL might be higher, and that the total annotation time (i.e., human labor) might be lower than traditional supervised learning.

AL does not directly affect the time saved in data collection. However, a model's uncertainty based on what it already knows about specific classes or scenarios, can be monitored and used to filter away incoming data, which may be noisy and unnecessary. It can instead be used to collect data that is more relevant and informative, thereby reduce the resources needed to hold large amounts of unimportant data.

**RQ2:** Does there exist an optimal query strategy that can be used for either object detection and/or segmentation in the setting of autonomous perception?

We implement multiple query strategies combined with various aggregation techniques to examine their behaviors when used with different CV tasks and datasets.

Our findings indicate that the strategies using SUM as the aggregation technique benefit from selecting more instance-rich images. In most experiments, except EXP 5, SUMENT has the highest performance compared to the other learners. In addition, it benefits from using No-Spectrum as a selection method, as explained in EXP 3.1. EXP 4 is on instance-based segmentation. DROPOUT performs better than the baseline, but its computational expensiveness does not reflect its performance; see RQ3 for further details.

It is important to mention that since we are working with black-box methods, finding and combining an optimal query strategy can be seen as a guesstimate. We simply select one, try it out, and try to understand how it behaves. As demonstrated by the experimental results, the performance change drastically when we use a real-life dataset compared to a synthetic and pre-processed dataset (Section 6.1).

We believe that having a query strategy that is able to collect highly informative images containing multiple objects is highly beneficial to be used in training, as demonstrated by our results. In addition, other techniques such as data diversity and sample selection methods might be beneficial to use with a query strategy.

A more extensive evaluation is needed to answer this question in more detail, and we can not conclude that there exists a specific query strategy that stands out for all tasks or that fits well in an autonomous setting. A combination of strategies to switch between may be useful for exploring and exploiting when needed, as presented by Bondu *et al.* [8]; for instance, early exploration and then later exploitation.

**RQ3:** Does AL perform differently depending on the CV task, or is there a similar performance pattern? Does it work well with instance-based segmentation?

The performance pattern is very similar on most experiments. However, strategies measure uncertainty differently depending on the CV task, and we notice that the complexity of the dataset has a higher impact on the performance, as explained in Section 6.1.

If AL works well with instance-based segmentation is hard to say. However, in EXP 4 (Section 5.4.4), DROPOUT is demonstrated to work for this task, as it performs better than the baseline learner and is comparable to the other learners. We calculate uncertainty using various metrics;

spatial mask uncertainty, spatial bounding box uncertainty, and semantic uncertainty. Due to the computationally heavy nature of DROPOUT, it is hard to recommend this strategy, as this does not reflect its performance. We believe that Morrison's *et al.* [54] approach has to be optimized using thresholds to neglect false positives, as explained in Section 3.3.1.

We can not conclude if AL works well with instance-based segmentation based on our results, since the performance during the early ALIs is heavily influenced by the usage of the correct aggregation strategy. Using DROPOUT might play a bigger role much later in the ALIs, or in combination with transfer learning (Section 5.6.4). Due to time constraints, this was set to be for future work.

## 6.7 Limitations

Our thesis clearly has some limitations. However, we believe our work could be a good starting point for further studies and development in NAPLab using Active Learning.

As a reminder, we do not focus on training and model optimization to achieve the "best" possible model. In our earlier experiments, the models might not have been given enough training time to understand the initial training set fully, or they might have been trained for too long. However, in EXP 5.2, this was confirmed by using early stopping; the learners needed more training.

We did not have the chance to try out all strategy combinations or different batch sizes, as this is a hard metric to tune. Most of our decisions are trial and error; some values are arbitrarily chosen, other values, decisions, and implementation choices are based on what worked in previous research approaches.

## 6.8 Reflection

Early in this thesis, the focus was on getting a broad overview of the field of AL, to then narrow it down into a specific problem. Later, the time went into performing smaller experiments, testing, debugging, and implementing the AL Framework.

Running active learning experiments took time, as the models had to be trained for multiple iterations. Tons of time was spent in training models for weeks, and then re-training due to unexpected errors or optimization purposes. A lot of time went into implementing the AL Framework, making it modular and scalable for future use; doing so accelerated my ability to perform new experiments and add new functionality.

If I were to work with this thesis again, I would have followed the same path as where I am now but reached it much quicker. With regard to

the knowledge I currently have, I would have been able to conduct more experiments and further investigate the usage of AL in the autonomous domain. Relevant experiments would have been: initial training using a more informative training set than selecting samples randomly, further investigating the explore vs. exploit trade-off by combining the proposed methods Spectrum and Data Diversity, further improving early stopping, and further optimizing the uncertainty measures for instance-based segmentation.

As a final note, I hope the work done in this thesis opens up possibilities for NAPLab for further research.

# Chapter 7

# Conclusion and Future Work

In this thesis, we explore Active Learning (AL) and evaluate its effectiveness on object detection, and instance-based segmentation in an autonomous domain. We propose a novel pool-based AL Framework that applies this strategy using state-of-the-art Faster-R-CNN and Mask R-CNN models on autonomous driving datasets such as Apollo Synthetic, Waymo Open, and data collected from Trondheim. Our results demonstrate that active learning outperforms the random selection baseline by selecting highly informative samples for training. Furthermore, these results indicate that AL reduces the amount of training data required and minimizes the annotation job.

Our AL Framework is modular and can be configured to run custom experiments. It includes various uncertainty-based active learners that are trained and compared to a baseline random-selection learner. Other interesting findings indicate that learners using SUM as aggregation technique have increased performance as they select instance-rich samples, learners using AVG have decreased performance when introduced to false positives, and transfer learning can be used to create an informative initial training set. In addition to these findings, we evaluate how different selection methods, unbalanced initial sets, data diversity, and early stopping affects the performance of AL.

NAPLab has a goal to develop and research state-of-the-art models using data from complex nordic environments. AL would be beneficial in such a scenario to quickly create labeled and information-rich datasets to make their models even more robust. We believe our contributions open up new possibilities for NAPLab to employ AL as a part of their future pipeline.

## 7.1 Future Work

Even though the results presented in this thesis are interesting and motivating, it is still considered as a starting point, and further evaluation is needed. As for the near future, we have ambitions to publish our work as a paper and have already started our preparations.

We will now briefly discuss some techniques and ideas that might be worth looking into as future work.

### 7.1.1 AL Pipeline

We suggest a novel machine learning pipeline, which fits our AL Framework, to be used in an autonomous setting as future work. There does not exist a specific set of steps in a pipeline that must be followed. Various pipelines are made for different purposes, and the structure can vary depending on the specific needs. The steps of our proposed pipeline are as follows (See Figure 7.1 for an illustration):

1. **Collect Sensor Data:** Collect raw data using the various sensors (e.g., Camera, LIDAR, RADAR) that are located on a vehicle. Synchronize, compress, and organize the raw data in a meaningful way directly on the vehicle.

2. **Clean Raw Sensor Data:** Remove noisy, corrupted, duplicate, and unnecessary data in order to save storage space. Ensure data consistency and create a dataset.

3. **Data Anonymization:** Send the data through an anonymization process (e.g., DeepPrivacy (Hukkelås *et al.* [34])). If possible, perform this directly on the vehicle to ensure GDPR compliance.

4. **Secure Data Transportation:** Move the data from the vehicle to a server in a secure way using encryption.

5. **Extract Data:** Extract a small set of randomly selected samples from the fully pre-processed unlabeled set.

6. **Annotate Data:** Start annotating the obtained set manually to create an initial training set. Speed up the annotation process by using popular CV annotation tools such as CVAT [56] or VoTT [52].

7. **Initiate AL:** Initiate the AL process using the initial training set with the AL Framework to obtain a new set of highly informative samples. The framework is responsible for training, saving, and evaluating models. Samples having high informativeness should be manually annotated. Samples having low informativeness indicate high model confidence, and can, therefore, be automatically annotated using their

predictions, with human supervision and quality control. Samples with medium informativeness can be ignored. Repeat Steps 6 and 7 until a stopping criteria has been reached.

8. **Deploy and Monitor Model:** Deploy the fully trained model after AL and monitor it to save performance logs.

### Co-Operative Learning

With co-operative learning, both a model and a human annotator can be used for annotating samples. Over time, a model using AL will become better at differentiating between samples with high and low informativeness. Samples having low informativeness contain detections with low informativeness. These detections can either be used to automatically annotate images or give annotation suggestions that can be approved by a human annotator for quality assurance. Doing so can save more time in annotating and make the labeling process even faster. For instance, by providing a user interface, the annotator can be given highly informative images one by one to label, and highly certain images to edit and verify.

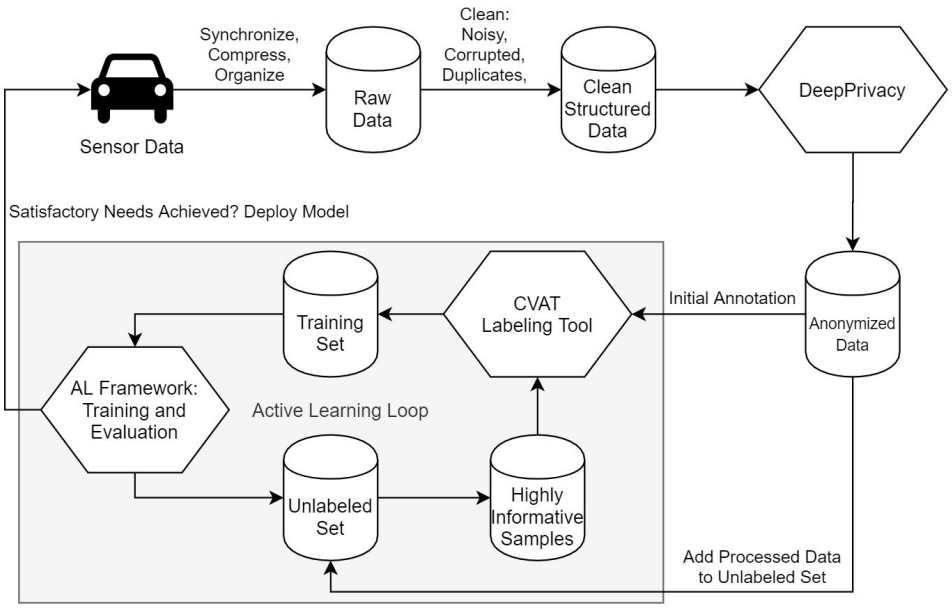



**Figure 7.1:** A simple illustration of our proposed AL pipeline. Raw sensor data is collected from a vehicle, processed and anonymized (e.g., DeepPrivacy [34]). An initial random set of samples are annotated using a annotation tool (e.g., CVAT [56]), and the active learning loop is initiated. When a model has been fully trained with AL, it is deployed to a vehicle.

### 7.1.2 Instance-Based Segmentation

To our knowledge, few approaches have been applying AL on instance-based segmentation. Our attempt with using the proposed method by Morrison *et al.* [54], gave promising results. However, this method could not be directly compared to any relevant work. Nevertheless, we believe this method can be further improved.

### 7.1.3 White-Box Query Strategies

A white-box query strategy can be used to make more complex measurements by considering other metrics than just the output of the model. However, this requires that you understand the model's architecture. Using a white-box query strategy had a better overall performance compared to black-box query strategies, as seen in Roy *et al.* [66]. Kao *et al.* [39] measures the localization tightness based on the IoU between the regional proposal and the refined bounding box in Faster R-CNN; thus, it can be seen as a white-box method. Recently, Zhang *et al.* [87] proposes Mask-Refined R-CNN, which improves the segmentation accuracy compared to Mask R-CNN. They present a refinement framework that is the core of their proposed algorithm. A similar white-box approach could be to measure the uncertainty based on the mask refinement; the more refinement needed, the higher uncertainty of the model.

### 7.1.4 Representativeness

When selecting informative samples, several measurements can be made, Huang *et al.* [33] combines both the informativeness and representativeness of a sample. Representativeness measured if samples represent the input pattern of unlabeled data well (Huang *et al.* [33]). Their work is based on binary classification, but claim that it can be developed for multi-class problems. In addition., they mention other works that use similarity metrics to find representative samples.

### 7.1.5 Balanced Training Sets

Methods can be added to prevent unbalanced training sets. Brust *et al.* [9] counts minority and majority classes to make sure there is a balance when selecting new, highly informative samples. Ertekin *et al.* [16] selects equal amounts of highly informative samples from each class that are closer to the hyperplane. Their proposal is heavily based on SVMs.

### 7.1.6 Using Probabilistic Networks to Measure Uncertainty

AL is tightly coupled to measuring the uncertainty of a model. Gast's *et al.* [24] approach is not directly related to AL, but they measure the uncer-

tainty of a network using a lightweight probabilistic deep neural network. They replace the output layers with probabilistic output layers, and the intermediate activations by distributions. Measuring the uncertainty based on a model's predictions using softmax might not be a good measure, as explained in Section 6.2.

# Bibliography


[1] A. Kramida et al. NIST Atomic Spectra Database (ver. 5.7.1), [Online]. Available: https://physics.nist.gov/asd [2017, April 9]. National Institute of Standards and Technology, Gaithersburg, MD. 2019.

[2] Waleed Abdulla. *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. https://github.com/matterport/Mask_RCNN. 2017.

[3] Yotam Abramson and Yoav Freund. "Active learning for visual object detection". In: *UCSD Technical Report* (Jan. 2006).

[4] Hamed H. Aghdam et al. *Active Learning for Deep Detection Neural Networks*. 2019. arXiv: 1911.09168 [cs.CV].

[5] Abhijit Bendale and Terrance Boult. *Towards Open Set Deep Networks*. 2015. arXiv: 1511.06233 [cs.CV].

[6] Alberto Bietti. *Active Learning for Object Detection on Satellite Images*. 2012.

[7] Daniel Bolya et al. *YOLACT: Real-time Instance Segmentation*. 2019. arXiv: 1904.02689 [cs.CV].

[8] A. Bondu, V. Lemaire, and M. Boullé. "Exploration vs. exploitation in active learning : A Bayesian approach". In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. July 2010, pp. 1–7. DOI: 10.1109/IJCNN.2010.5596815.

[9] Clemens-Alexander Brust, Christoph Käding, and Joachim Denzler. *Active Learning for Deep Object Detection*. 2018. arXiv: 1809.09875 [cs.CV].

[10] H. Caesar, J. Uijlings, and V. Ferrari. "COCO-Stuff: Thing and Stuff Classes in Context". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018, pp. 1209–1218. DOI: 10.1109/CVPR.2018.00132.

[11] *Common Objects in Context (COCO), Detection evaluation metrics*. URL: http://cocodataset.org/#detection-eval.

[12] Tivadar Danka and Peter Horvath. "modAL: A modular active learning framework for Python". In: *GitHub repository* (2018). available on arXiv at https://arxiv.org/abs/1805.00979, and GitHub at https://github.com/modAL-python/modAL. arXiv: 1805.00979 [cs.LG].

[13] J. Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[14] E. D. Dickmanns. "The development of machine vision for road vehicles in the last decade". In: *Intelligent Vehicle Symposium, 2002. IEEE*. Vol. 1. 2002, 268–281 vol.1.

[15] Melanie Ducoffe and Frederic Precioso. "Active learning strategy for CNN combining batchwise Dropout and Query-By-Committee". In: *ES2017-122 ESANN*. 2017.

[16] Seyda Ertekin et al. "Learning on the Border: Active Learning in Imbalanced Data Classification". In: *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*. CIKM '07. Lisbon, Portugal: Association for Computing Machinery, 2007, pp. 127–136. ISBN: 9781595938039. DOI: 10.1145/1321440.1321461. URL: https://doi.org/10.1145/1321440.1321461.

[17] Mark Everingham et al. "The Pascal Visual Object Classes Challenge: A Retrospective". In: *International Journal of Computer Vision* 111.1 (2015), pp. 98–136. ISSN: 1573-1405. DOI: 10.1007/s11263-014-0733-5. URL: https://doi.org/10.1007/s11263-014-0733-5.

[18] Yoav Freund and Robert E. Schapire. "A desicion-theoretic generalization of on-line learning and an application to boosting". In: *Computational Learning Theory*. Ed. by Paul Vitányi. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 23–37. ISBN: 978-3-540-49195-8.

[19] L. Fridman et al. "MIT Advanced Vehicle Technology Study: Large-Scale Naturalistic Driving Study of Driver Behavior and Interaction With Automation". In: *IEEE Access* 7 (2019), pp. 102021–102038. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2926040.

[20] Yarin Gal and Zoubin Ghahramani. *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. 2015. arXiv: 1506.02142 [stat.ML].

[21] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. *Deep Bayesian Active Learning with Image Data*. 2017. arXiv: 1703.02910 [cs.LG].

[22] J. Gall et al. "Hough Forests for Object Detection, Tracking, and Action Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.11 (2011), pp. 2188–2202.

[23] J. Gast and S. Roth. "Lightweight Probabilistic Deep Networks". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3369–3378.

[24] Jochen Gast and Stefan Roth. *Lightweight Probabilistic Deep Networks*. 2018. arXiv: 1805.11327 [cs.CV].

[25] R. Girshick. "Fast R-CNN". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448.

[26] Ross Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2013. arXiv: 1311.2524 [cs.CV].

[27] Z. Gong, P. Zhong, and W. Hu. "Diversity in Machine Learning". In: *IEEE Access* 7 (2019), pp. 64323–64350. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2917620.

[28] Chuan Guo et al. *On Calibration of Modern Neural Networks*. 2017. arXiv: 1706.04599 [cs.LG].

[29] David Hall et al. *Probabilistic Object Detection: Definition and Evaluation*. 2018. arXiv: 1811.10800 [cs.CV].

[30] K. He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[31] K. He et al. "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[32] A. Holub, P. Perona, and M. C. Burl. "Entropy-based active learning for object recognition". In: *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. June 2008, pp. 1–8. DOI: 10.1109/CVPRW.2008.4563068.

[33] S. Huang, R. Jin, and Z. Zhou. "Active Learning by Querying Informative and Representative Examples". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.10 (2014), pp. 1936–1949.

[34] Håkon Hukkelås, Rudolf Mester, and Frank Lindseth. *DeepPrivacy: A Generative Adversarial Network for Face Anonymization*. 2019. arXiv: 1909.04538 [cs.CV].

[35] P. Jain and A. Kapoor. "Active learning for large multi-class problems". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, pp. 762–769.

[36] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. "Multi-class active learning for image classification". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2009, pp. 2372–2379. DOI: 10.1109/CVPR.2009.5206627.

[37] D. Kale and Y. Liu. "Accelerating Active Learning with Transfer Learning". In: *2013 IEEE 13th International Conference on Data Mining*. 2013, pp. 1085–1090.

[38] Jaeho Kang, Kwang Ryel Ryu, and Hyuk-Chul Kwon. "Using Cluster-Based Sampling to Select Initial Training Set for Active Learning in Text Classification". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Honghua Dai, Ramakrishnan Srikant, and Chengqi Zhang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 384–388. ISBN: 978-3-540-24775-3.

[39] Chieh-Chi Kao et al. "Localization-Aware Active Learning for Object Detection". In: *ACCV*. 2018. arXiv: 1801.05124 [cs.CV].

[40] Ashish Kapoor et al. "Gaussian Processes for Object Categorization". In: *International Journal of Computer Vision* 88.2 (2010), pp. 169–188. ISSN: 1573-1405. DOI: 10.1007/s11263-009-0268-3. URL: https://doi.org/10.1007/s11263-009-0268-3.

[41] Keras-Team. *MNIST CNN - Example*. https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py. 2019.

[42] A. Kirillov et al. "Panoptic Segmentation". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019, pp. 9396–9405. DOI: 10.1109/CVPR.2019.00963.

[43] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: *University of Toronto* (2009).

[44] Yann LeCun and Corinna Cortes. *MNIST handwritten digit database*. http://yann.lecun.com/exdb/mnist/. 2010. URL: http://yann.lecun.com/exdb/mnist/.

[45] Fei-Fei Li, Marco Andreetto, and Marc 'Aurelio Ranzato. "Caltech101 Image Dataset". In: (2003). URL: http://www.vision.caltech.edu/Image_Datasets/Caltech101/.

[46] X. Li and Y. Guo. "Adaptive Active Learning for Image Classification". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2013, pp. 859–866. DOI: 10.1109/CVPR.2013.116.

[47] T. Lin et al. "Feature Pyramid Networks for Object Detection". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017, pp. 936–944. DOI: 10.1109/CVPR.2017.106.

[48] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2014. arXiv: 1405.0312 [cs.CV].

[49] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *Lecture Notes in Computer Science* (2016), pp. 21–37. ISSN: 1611-3349. DOI: 10.1007/978-3-319-46448-0_2. URL: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[50] C. Long and G. Hua. "Multi-class Multi-annotator Active Learning with Robust Gaussian Process for Visual Recognition". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015, pp. 2839–2847. DOI: 10.1109/ICCV.2015.325.

[51] Radek Mackowiak et al. *CEREALS - Cost-Effective REgion-based Active Learning for Semantic Segmentation*. Oct. 2018.

[52] Microsoft. *microsoft/VoTT*. https://github.com/microsoft/VoTT. May 2020.

[53] D. Miller et al. "Evaluating Merging Strategies for Sampling-based Uncertainty Techniques in Object Detection". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 2348–2354.

[54] Doug Morrison, Anton Milan, and Epameinondas Antonakos. "Uncertainty-aware Instance Segmentation using Dropout Sampling". In: *Computer Vision and Pattern Recognition (CVPR)*. 2019.

[55] Keiron O'Shea and Ryan Nash. *An Introduction to Convolutional Neural Networks*. 2015. arXiv: 1511.08458 [cs.NE].

[56] Opencv. *opencv/cvat*. https://github.com/opencv/cvat. May 2020.

[57] S. J. Pan and Q. Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (Oct. 2010), pp. 1345–1359. ISSN: 2326-3865. DOI: 10.1109/TKDE.2009.191.

[58] Adam Paszke et al. *ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation*. 2016. arXiv: 1606.02147 [cs.CV].

[59] Luis Perez and Jason Wang. *The Effectiveness of Data Augmentation in Image Classification using Deep Learning*. 2017. arXiv: 1712.04621 [cs.CV].

[60] Dean A. Pomerleau. "ALVINN: An Autonomous Land Vehicle in a Neural Network". In: *Advances in Neural Information Processing Systems 1*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 305–313. ISBN: 1558600159.

[61] Lutz Prechelt. "Early Stopping-But When?" In: *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*. Berlin, Heidelberg: Springer-Verlag, 1998, pp. 55–69. ISBN: 3540653112.

[62] Zhenshen Qu et al. *Deep Active Learning for Remote Sensing Object Detection*. 2020. arXiv: 2003.08793 [cs.CV].

[63] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *ArXiv* abs/1804.02767 (2018).

[64] S. Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (June 2017), pp. 1137–1149. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2016.2577031.

[65] Soumya Roy, Vinay P. Namboodiri, and Arijit Biswas. *Active learning with version spaces for object detection*. 2016. arXiv: 1611.07285 [cs.CV].

[66] Soumya Roy, Asim Unmesh, and Vinay P. Namboodiri. "Deep active learning for object detection". In: *British Machine Vision Conference (BMVC)*. 2018.

[67] Ozan Sener and Silvio Savarese. *Active Learning for Convolutional Neural Networks: A Core-Set Approach*. 2017. arXiv: 1708.00489 [stat.ML].

[68] B. Settles. *Active Learning Literature Survey*. Computer Sciences Technical Report 1648. University of Wisconsin–Madison, 2009.

[69] B. Settles, M. Craven, and L. Friedland. "Active Learning with Real Annotation Costs". In: *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*. 2008.

[70] Connor Shorten and Taghi M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning". In: *Journal of Big Data* 6.1 (July 2019), p. 60. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0.

[71] Michael Sörsäter. *Active Learning for Road Segmentation using Convolutional Neural Networks*. Master's thesis. Linköping University, 2018.

[72] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 1929–1958. ISSN: 1532-4435.

[73] Pei Sun et al. *Scalability in Perception for Autonomous Driving: Waymo Open Dataset*. 2019. arXiv: 1912.04838 [cs.CV].

[74] Baidu Apollo team. *Apollo Synthetic - Photo-Realistic Dataset for Autonomous Driving*. http://apollo.auto/synthetic.html. 2019.

[75] Ub. *USPS dataset*. https://www.kaggle.com/bistaumanga/usps-dataset. Apr. 2018.

[76] A. Vezhnevets, J. M. Buhmann, and V. Ferrari. "Active learning for semantic segmentation with expected change". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2012, pp. 3162–3169. DOI: 10.1109/CVPR.2012.6248050.

[77] S. Vijayanarasimhan and K. Grauman. "Large-scale live active learning: Training object detectors with crawled data and crowds". In: *CVPR 2011*. 2011, pp. 1449–1456.

[78] Sudheendra Vijayanarasimhan and Kristen Grauman. "Cost-Sensitive Active Visual Category Learning". In: *International Journal of Computer Vision* 91 (2010), pp. 24–44.

[79] K. Wang et al. "Cost-Effective Active Learning for Deep Image Classification". In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.12 (Dec. 2017), pp. 2591–2600. ISSN: 1558-2205. DOI: 10.1109/TCSVT.2016.2589879.

[80] *Waymo Open Dataset: An autonomous driving dataset*. https://www.waymo.com/open. 2019.

[81] Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. "Probability Estimates for Multi-Class Classification by Pairwise Coupling". In: *J. Mach. Learn. Res.* 5 (Dec. 2004), pp. 975–1005. ISSN: 1532-4435.

[82] Yuxin Wu et al. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.

[83] Lin Yang et al. *Suggestive Annotation: A Deep Active Learning Framework for Biomedical Image Segmentation*. 2017. arXiv: 1706.04737 [cs.CV].

[84] A. Yao et al. "Interactive object detection". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, pp. 3242–3249.

[85] YunYang1994. *YYMNIST - A MNIST Dataset for Object Detection*. https://github.com/YunYang1994/yymnist. 2019.

[86] Richard Zhang et al. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In: *CVPR*. 2018.

[87] Yiqing Zhang et al. "Mask-Refined R-CNN: A Network for Refining Object Details in Instance Segmentation". eng. In: *Sensors (Basel, Switzerland)* 20.4 (Feb. 2020). s20041010[PII], p. 1010. ISSN: 1424-8220. DOI: 10.3390/s20041010. URL: https://doi.org/10.3390/s20041010.

[88] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (Apr. 2004), pp. 600–612. ISSN: 1941-0042. DOI: 10.1109/TIP.2003.819861.

# Appendix A

# AL Framework

## A.1 AL Framework Architecture

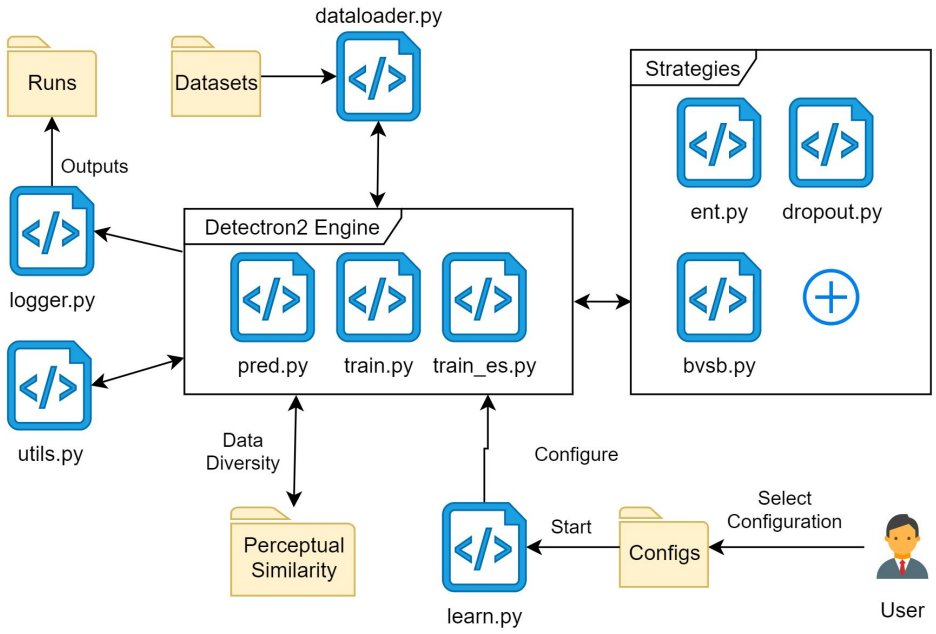



**Figure A.1:** This figure illustrates the architecture of the AL Framework. The user provides a configuration file from the configs folder to learn.py to run experiments. The AL engine uses Detectron2, and contains training and predictions loops. The AL engine uses different Query Strategies and DataLoaders to perform experiments. All logs and outputs are saved in the run folder. Other utility functions are accessed from utils.py. A Perceptual Similarity tool is used for data diversity. The code of this AL Framework can be downloaded from Google Drive.

## A.2 Setup

**Software:** We use Detectron2 as the object detection and instance segmentation framework. In EXP 1, we use ModAL, which is a modular active learning framework for Python3. We use Google Colab quick testing and debugging purposes.

**Hardware:** We use two Tesla V100 GPU's located on the NAP Server with 32 GB RAM each.

Models used from Detectron2's model zoo [82]:

- COCO-pretrained R50-FPN 3x schedule Faster R-CNN Model for Object Detection (Model ID: 137849458)
- COCO-pretrained R50 FPN 3x schedule Mask R-CNN Model for Object Detection and Instance-Based Segmentation (Model ID: 137849600)

## A.3 Detection Evaluation Metrics

To evaluate the different learners on a test set, we a DatsetEvaluator from Detectron2. The DatasetEvaluator follows the detection evaluation metrics used by COCO [11]. We present the list of metrics given as output from the DatasetEvaluator below. Keep in mind, we only use AP50 to compare the performances of each learner.
Description of each metric. See COCO's website for more details.

- AP - Average Precision at IoU=.50:.05:.95
- **AP50 - Average Precision at IoU=.50**
- AP75 - Average Precision at IoU=.75
- APs - Average Precision for small objects: area ¡ $32^2$
- APm - Average Precision for medium objects: $32^2$ ¡ area ¡ $96^2$
- APl - Average Precision for large objects: area ¿ $96^2$
- AP-<class> - Average Precision for each class at IoU=.50:.05:.95

# Appendix B

# Query Strategy Algorithms

**Algorithm 1:** Learners: SUMENT, MAXENT, and AVGENT

```
Input: images, k
Output: image_scores[:k]
image_scores = [];
for each image in images do
    detection_scores = [];
    for each detection in image do
        filename = detection["filename"];
        softmax = detection["softmax"];
        # Calculate entropy of softmax;
        entropy = Entropy(softmax);
        detection_scores.append(entropy);
    end
    # Use the chosen aggregation technique;
    image_score = MAX(detection_scores);
    image_score = SUM(detection_scores);
    image_score = AVG(detection_scores);
    image_scores.append([filename, image_score]);
end
# Highest scores first;
return sorted(image_scores)[:k];
```

**Algorithm 2:** Learners: SUMBVSB, MAXBVSB, and AVGBVSB

```
Input: images, k
Output: image_scores[:k]
image_scores = [];
for each image in images do
    detection_scores = [];
    for each detection in image do
        filename = detection["filename"];
        softmax = detection["softmax"];
        # Get the best and second-best probability;
        best_prob = MAX(softmax);
        second_best_prob = SECONDMAX(softmax);
        margin = best_prob - second_best_prob;
        detection_scores.append(1-margin);
    end
    # Use the chosen aggregation technique;
    image_score = MAX(detection_scores);
    image_score = SUM(detection_scores);
    image_score = AVG(detection_scores);
    image_scores.append([filename, image_score]);
end
# Highest scores first;
return sorted(image_scores)[:k];
```

**Algorithm 3:** Active Learner: DROPOUT

```
Input: images, k
Output: image_scores[:k]
image_scores = [];
for each image_set in images do
    observations = {};
    threshold = 0.4;
    # image_set contains the inference run over the same image i
     times using dropout;
    for each image in image_set do
        groups = {};
        for each detection in image do
            # Create groups of detections having IOU > threshold;
        end
        # Add each group to observations;
    end
    sum_uncertainty = 0;
    for each observation in observations do
        mean_softmax = mean(observation.softmaxes);
        mean_bbox = mean(observation.bboxes);
        mean_mask = mean(observation.masks);
        mask_IOUs = IOU(mean_mask, observation.masks);
        bbox_IOUs = IOU(mean_bbox, observation.bboxes);
        u_sem = max(mean_softmax) - second_max(mean_softmax);
        u_spl_mask = mean(mask_IOUs);
        u_spl_bbox = mean(bbox_IOUs);
        u_n = detection appearance out of i inferences;
        sum_uncertainty +=
          1.0-(u_sem*u_spl_mask*u_spl_bbox*u_n);
    end
    image_scores.append([filename, sum_uncertainty]);
end
# Highest scores first;
return sorted(image_scores)[:k];
```

# Appendix C

# Issues

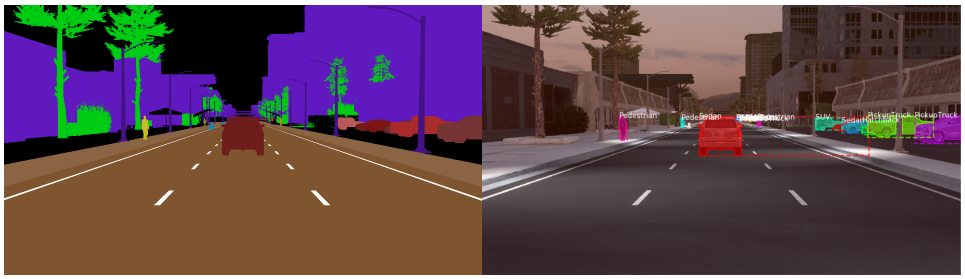
This is a more detailed explanation of the various issues we encountered:

## C.1 Matterport's Mask R-CNN Implementation

- As mentioned in Section 3.2.1, we started with Single Shot Multibox Detector (SSD) for object detection, and Matterport's Mask R-CNN [2] for both object detection and segmentation. While still trying to solve these issues, the PyTorch library Detectron2 [82] was released. Due to time constraints and persisting problems (See the next point for details), we decided to use our framework with Detectron2 instead. In addition, other projects in NAPLab are using this framework too, which makes our AL Framework compatible to be used for future work.
- The values from the RPN layers were giving a highly fluctuating loss on the validation sets. This did not just happen on the Apollo Synthetic dataset but also the already existing sample sets. We tried to solve this issue but with no success. These highly fluctuating losses made our results unreliable and gave high variation in average precision values. If we compare this with Detectron2, Detectron2 was much more stable.

## C.2 Apollo Synthetic Dataset

- Each "Traffic_XYZ" folder contains files with id-names 0 to N (000000.jpg, 000001.jpg). This gave us multiple files having the same ID. We transformed the hierarchical structure into stream-based and gave each image a unique ID. The following four folders were made: RGB - .jpg files, SEG - segmented .png files, ENC - encodings for each SEG file .txt and OBJ - object ground truth files .txt.
- The instance segmentation encoding files from image 0030220.jpg to image 0030229.jpg (possibly more) contains the same color encodings



**(a)** Segmented Ground Truth **(b)** SUV and Sedan as a single segment

for both the Sedan and SUV classes. This encoding does not make them unique, since the segment of an SUV has the same color as a Sedan. Figure C.1b shows that two different class instances are seen as one due to the matching color. To explain the process: The color encoding for each class instance is used to generate a binary mask, which represents that instance; in order to differentiate class instances in an image, each color encoding is required to be unique. Since this requirement is not met in some images, instances from different classes get grouped. This issue was resolved by removing these images that had duplicate color encodings.

- Some images have ground truths of non-existing objects.

Navjot Singh

# Deep Active Learning for Autonomous Perception

NTNU
Kunnskap for en bedre verden