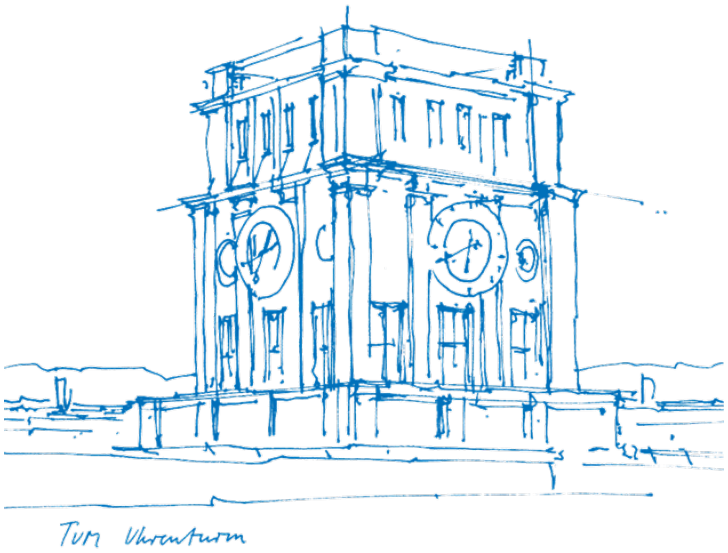
TUM School of Computation, Information and Technology
Department of Electrical and Computer Engineering
Technical University of Munich

Real-Time Object Detection Uncertainty Quantification using Augmented Images for Autonomous Vehicles

Unsicherheitsquantifizierung der Objekterkennung in Echtzeit mithilfe erweiterter Bilder für autonome Fahrzeuge

Mostafa ElHayani



Real-Time Object Detection Uncertainty Quantification using Augmented Images for Autonomous Vehicles

Unsicherheitsquantifizierung der Objekterkennung in Echtzeit mithilfe erweiterter Bilder für autonome Fahrzeuge

Mostafa ElHayani

TUM School of Computation, Information and Technology
Department of Electrical and Computer Engineering
Technical University of Munich

Real-Time Object Detection Uncertainty Quantification using Augmented Images for Autonomous Vehicles

Unsicherheitsquantifizierung der Objekterkennung in Echtzeit mithilfe erweiterter Bilder für autonome Fahrzeuge

Mostafa ElHayani

Thesis for the attainment of the academic degree

Master of Science (M.Sc.)

at the Department of Electrical and Computer Engineering of the Technical University of Munich.

Examiner:
Prof. Dr. Hans-Joachim Bungartz

Supervisors:
Kislaya Ravi (TUM), Dr. Liang Yu (CARIAD), Shubham Khatri (CARIAD)

Submitted:
Munich, 15.02.2024

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Munich, 15.02.2024 Mostafa ElHayani

Abstract

Uncertainty quantification (UQ) is crucial in developing and deploying autonomous driving systems. Autonomous vehicless (AVs) operate in complex and dynamic environments where uncertainties in road conditions, traffic scenarios, and sensor limitations can significantly impact their performance and safety. Accurately quantifying and managing uncertainty ensures reliable decision-making and mitigating risks associated with autonomous driving. For AVs, it is essential to develop efficient solutions regarding memory and computation requirements. AVs operate in real-time, dynamic environments where quick and accurate decision-making is critical. Therefore, UQ methods must balance providing reliable uncertainty estimates and being computationally lightweight. We propose a new method to address some shortcomings of previous approaches, reducing the computational overhead of doing multiple non-parallelizable samplings and the memory usage of having various models by generating numerous predictions using multi-model sampling techniques. Test-time augmentations can be applied to the input images at inference time to produce diverse outputs that are still plausible according to the learned distribution. They can be used instead of maintaining a large number of models or having to run multiple dropout samplings. The work in this thesis investigates the usage of augmentations, such as noise, blurring, or pixel value manipulations, on the input image as a batch prediction and using the output to estimate the uncertainty of the model. The proposed approach is tested on Yolov5 trained on the Mapillary dataset. Furthermore, we demonstrate the efficacy of the proposed algorithm in an active learning experiment. We further compare our method with two state-of-the-art techniques: deep ensembles and MC-Dropout.

Contents

1 Introduction

1.1 Motivation

In the recent years, there have been great advances in artificial intelligence (AI) [1, 2, 3]. The enhanced capabilities of deep learning (DL) have significantly improved the precision of neural networks (NNs). Consequently, these networks are now relied upon to make intricate decisions in various applications, including, but not limited to, autonomous driving [4], speech recognition [5], and medical diagnosis [6]. Within these specific domains, NNs play a crucial role as indispensable components within larger decision-making pipelines. Their heightened accuracy has paved the way for their widespread use and integration into diverse fields, marking a notable advancement in the field of AI.

Research in recent years has shown that with the precision and accuracy advancements of modern NNs and machine learning (ML) models, there has also been an unwanted side effect of unreliability and increased uncertainty [7, 8]. Accuracy measures the correctness of predictions, representing the ratio of correct classifications to the total dataset. Certainty, on the other hand, gauges the confidence level in the model's predictions. While Accuracy focuses on correctness, Certainty indicates the model's classification assurance. NNs were shown to produce overconfident predictions even on unseen out-of-distribution data points [9], as well as, irrelevant inputs [10]. The effectiveness and robustness of such models extend beyond their accuracy metrics. It's also necessary to tell when the model is unsure of its prediction. This ensures a more transparent and reliable decision-making process, acknowledging the inherent uncertainties in real-world scenarios or those stemming from the finite nature of the available training data, which fails to encompass all possible scenarios. For autonomous vehicles (AVs), where NNs are employed for navigation and object detection (OD) [11], it is imperative that the system can recognize and communicate its uncertainty during ambiguous or challenging situations, enabling the vehicle to prioritize safety. Similar applications include healthcare applications where NNs should not only provide accurate assessments but also indicate the reliability of the predictions. This allows healthcare professionals to exercise caution and seek additional confirmation in the case of unreliable predictions.

In both examples, the incorporation of uncertainty awareness enhances the overall trustworthiness and safety of the decision-making systems that leverage NNs This thesis focuses on autonomous driving systems as a use case. However, the methods explained can also be used further for other systems.

Uncertainty quantification (UQ) [12, 13] is crucial in developing and deploying autonomous driving systems. AVs navigate intricate and ever-changing environments characterized by uncertainties in the road conditions, traffic scenarios, and limitations inherent in their sensor technologies. The multifaceted nature of these dynamic settings poses substantial challenges to the performance and safety of AVs. To foster dependable decision-making and alleviate potential risks associated with autonomous driving, it becomes imperative to precisely gauge and effectively manage the uncertainties intertwined with the operation of these vehicles. Acknowledging the uncertainties arising from unpredictable road conditions and sensor constraints and developing strategies to mitigate these challenges are paramount. Efficiently addressing uncertainties is vital for establishing a strong foundation in AV technology, enhancing reliability and safety in real-world situations. In dynamic environments that demand swift and accurate decision-making. UQ methods must find an equilibrium between delivering dependable estimates and maintaining computational efficiency in memory and computation solutions.

This thesis investigates one approach for UQ in AVs by employing the use of test-time augmentations (TTAs)[14, 15]. TTA refers to the application of data augmentation techniques during the testing or inference phases of ML models. Generating different input images using TTA alleviates the need to use different models or predict different times to generate an expression of probability distributions over model

parameters or predictions. This thesis hypothesizes that employing different augmentations at test time can yield prediction results comparable to the performance of alternative UQ methods. Using augmentations and assessing their impact on the models' predicted probabilities can provide a versatile and efficient means of uncertainty estimation. This hypothesis merits validation through experimentation, offering insights into the potential of TTA as an alternative approach for UQ in ML models.

1.2 Thesis structure

This thesis is divided into six chapters. Chapter 2 explains the necessary background knowledge needed in this thesis. In Chapter 3, we provide a comprehensive literature review. Chapter 4 discusses the methodology of the proposed approach and how the approach was tested. Chapter 5 explains our approach's uncertainty estimates, performance, and speed results compared to state of the art (SOTA) methods. Finally, Chapter 6 concludes this thesis, discussing the limitations and possible work that could be done in the future.

2 Background

2.1 Object Detection (OD)

This section aims to provide a comprehensive introduction to OD and delve into the specifics of the You only look once (YOLO) model. OD is a crucial computer vision task that involves identifying and localizing objects within images or videos. Unlike image classification, where a model assigns a single label to an entire image, or Segmentation, which assigns a label to each pixel, OD aims to locate and label multiple objects within an image, specifying their positions. This enables machines to recognize what's in an image and where those objects are located.

2.1.1 Visual object detection

The current landscape of visual OD is predominantly shaped by the dominance of deep neural networks (DNNs). The pivotal breakthrough in 2014 marked the advent of R-CNN [16, 17], which employed region proposals as inputs to a deep convolutional neural network classifier, namely AlexNet [18, 19], for accurate object localization. Subsequent advancements led to introducing Faster R-CNN [20, 21], enhancing training and testing speed by incorporating region proposal generation as an integral part of the network. More recently, the Single Shot Detector (SSD) [22] elevated this paradigm by unifying detection and proposal generation within a single network branch. This innovation allowed the detector to consider diverse image regions of varying sizes and resolutions, enhancing its adaptability.

2.1.2 YOLO

In parallel, YOLO is a groundbreaking object detection algorithm that significantly accelerated and improved object detection by introducing a unified framework that simultaneously predicts the bounding boxes and class probabilities. This streamlines the detection process and demonstrates notable efficacy in real-time applications. We explain the functioning of YOLO in the subsequent subsections.

Grid-based Approach

YOLO divides the input image into a grid of cells. Each cell is responsible for predicting objects that fall within its boundaries. This grid structure allows YOLO to detect objects across the entire image simultaneously.

Bounding box and class prediction

For each cell that contains an object, YOLO predicts a bounding box that tightly encloses the object. A bounding box is defined by the (x, y) coordinates of the box's center, width (w), and height (h). In addition to the bounding box, YOLO predicts the class of the object present in each cell. This involves assigning a probability to each class label from a predefined set of classes.

Each bounding box prediction comes with a confidence score that indicates the model's confidence in the accuracy of the bounding box and class prediction. This score helps filter out low-confidence detections.

Non-Maximum Suppression

After predictions, multiple overlapping bounding boxes might exist for the same object. Non-maximum suppression [23, 24] eliminates duplicate or highly overlapping boxes, retaining only the most confident one.

YOLO versions

Over time, YOLO has evolved with several versions, each improving upon its predecessor in accuracy and speed. Some notable versions include YOLOv2, YOLOv3, and YOLOv4, each introducing architectural enhancements [25, 26, 27]. The work in this thesis has been implemented using YOLOv5 found in this repository [28], as it's been proven to provide the best tradeoff between speed and accuracy [29, 30].

2.2 Uncertainty Quantification

Uncertainty is an inherent aspect of any decision-making process and reflects the lack of perfect knowledge. Understanding and quantifying uncertainty is crucial for making informed decisions and managing risks in different fields. Previous research [31, 32] has divided uncertainty in Bayesian modeling into two primary forms: aleatoric and epistemic uncertainties. Aleatoric uncertainty includes noise intrinsic to observations, like motion or sensor noise, that persists even when more data are collected. On the other hand, epistemic uncertainty arises from model parameters and can be reduced by collecting enough data. Research further divides aleatoric uncertainty into heteroscedastic and homoscedastic categories, which differ based on the qualities of the input. In computer vision, heteroscedastic uncertainty is significant and changes with the quality of the input, with highly textured images producing more confident predictions. Methods such as statistical analysis, probabilistic modeling, and expert judgment quantify and characterize uncertainty in different domains. In the realm of ML, uncertainties arise due to various factors, including data variability, model complexity, and inherent limitations of algorithms. This section delves into the importance of quantifying uncertainty and the methods that facilitate a deeper understanding of uncertainty in ML models. Approaches like Bayesian methods [33, 34, 35], ensemble techniques [36, 37] and dropout-based methods such as Monte Carlo dropout [38, 39] aim to capture and quantify these uncertainties, providing valuable insights into the reliability of model predictions.

2.2.1 Types of Uncertainty

Epistemic uncertainty [40, 41, 42] refers to the inherent limitations in our knowledge and understanding of a particular subject or system. It arises from incomplete information, imprecise measurements, or the inherent complexity of certain phenomena. It can be reduced through improved data collection, enhanced models, and increased knowledge. Recognizing and addressing epistemic uncertainty is crucial in various fields. It highlights the need for ongoing research, refinement of models, and continuous learning to enhance our understanding of the world. Effectively managing epistemic uncertainty allows for more informed and robust decision-making, enabling individuals and organizations to navigate complex scenarios with a clearer understanding of the inherent limitations in their knowledge.

Aleatoric uncertainty [43, 44, 45] pertains to the inherent randomness and unpredictability of different phenomena or processes. It arises from inherent variability, chance events, or stochastic processes that cannot be precisely determined or controlled. Unlike epistemic uncertainty, which can be mitigated through improved knowledge and understanding, aleatoric uncertainty is often irreducible. In different fields, understanding and quantifying aleatoric uncertainty is crucial for risk management and optimal decision-making.

2.2.2 Uncertainty quantification in Machine learning

In ML, quantifying uncertainty is crucial for obtaining reliable predictions and understanding the limitations of model outputs. One common approach is Bayesian inference, where uncertainty is expressed by placing

probability distributions over model parameters. In a Bayesian neural network (BNN), this is achieved by assigning a prior distribution $P(\theta)$ over the weights θ and updating it with observed data through Bayes' theorem to obtain the posterior distribution $P(\theta|D)$, where D represents the training data. The predictive distribution for a new input x is then given by integrating over all possible model parameters:

$$P(y|x, D) = \int P(y|x, \theta)P(\theta|D)\, d\theta \tag{2.1}$$

This integral represents the model's epistemic uncertainty, capturing uncertainty about the underlying parameters. Another source of uncertainty, aleatoric uncertainty, is often modeled by introducing a parameterized likelihood function $P(y|x, \phi)$, where ϕ represents parameters specific to the data. In some cases, dropout-based techniques, such as MC-dropout (3.2), provide a practical approximation to Bayesian inference.

An approach for UQ in ML involves analyzing the softmax scores produced by the model. Given the raw logits z_i for each class i, the softmax function transforms these scores into probabilities $P(y_i)$ by exponentiating and normalizing them. The softmax probabilities reflect the model's confidence in its predictions. To quantify uncertainty, the softmax scores are used to compute the entropy of the predicted probability distribution. Entropy, denoted as H, measures the disorder or unpredictability in a probability distribution. For a model predicting class probabilities $P(y_i)$ for each class i, the entropy is calculated as:

$$H = -\sum_i P(y_i) \log(P(y_i)) \tag{2.2}$$

A high entropy value indicates a more uncertain prediction, where the model assigns probabilities more evenly across different classes. In contrast, low entropy corresponds to a more confident prediction, with one or a few classes having high probabilities. Therefore, entropy is a quantitative measure of the uncertainty associated with a model's predictions based on the softmax scores. It provides valuable insights into the diversity and spread of the model's confidence across different classes, aiding uncertainty quantification and decision-making.

In the context of MC-dropout or ensemble methods (3.3), UQ involves leveraging multiple predictions from different model instances to capture various sources of uncertainty. The total uncertainty (U_{total}) can be computed using the entropy of the expectation of the softmax scores:

$$H = -\sum_i \mathbb{E}_{\text{MC}}[P(y_i)] \cdot \log(\mathbb{E}_{\text{MC}}[P(y_i)]) \tag{2.3}$$

$\mathbb{E}_{\text{MC}}$ represents the expectation over MC dropout samples or ensemble members. This entropy characterizes the overall uncertainty, considering both aleatoric and epistemic uncertainties.

The epistemic uncertainty ($U_{\text{epistemic}}$), associated with the variability between different model instances, can be calculated as the expectation of the entropy:

$$U_{\text{epistemic}} = \mathbb{E}_{\text{MC}}[H] \tag{2.4}$$

The aleatoric uncertainty ($U_{\text{aleatoric}}$), related to the inherent uncertainty in the data, can be computed as the entropy of the expectation:

$$U_{\text{aleatoric}} = -\sum_i \mathbb{E}_{\text{MC}}[\mathbb{E}_{\text{data}}[P(y_i)]] \cdot \log(\mathbb{E}_{\text{MC}}[\mathbb{E}_{\text{data}}[P(y_i)]]) \tag{2.5}$$

Here, $\mathbb{E}_{\text{data}}$ represents the expectation over data samples. This term captures the uncertainty that remains even when considering multiple model instances.

These equations and the associated uncertainty decomposition offer a comprehensive view of the different sources contributing to the overall uncertainty in predictions when employing MC dropout or ensemble methods. Calculating epistemic and aleatoric uncertainty provides valuable insights for robust decision-making in various ML applications.

2.3 Active Learning

The fundamental idea of Active learning (AL) [46, 47] is to selectively choose data points that add the most information to the training data. This allows ML systems to perform better with fewer labeled training examples.

In many contemporary ML issues, where unlabeled data may be plentiful, but labels are hard to come by, expensive, or time-consuming to get, active learning (AL) is a highly motivated approach. To enhance model performance with the least amount of labeled data possible, AL is a methodology that deliberately chooses the most instructive examples for annotation from an unlabeled dataset. Its usefulness is in obtaining similar or better performance with lower annotation costs. Methods for quantifying uncertainty are essential for improving AL because they offer a consistent approach to measuring the degree of uncertainty in model predictions. When a model has uncertainty estimates, it can be used to pinpoint situations in which it is unsure or prone to errors. The learning algorithm can improve its comprehension of unclear areas in the feature space or complex decision boundaries by actively choosing these uncertain samples for annotation. Techniques such as query-by-committee [48, 49], which selects instances where the model's predictions diverge among an ensemble of models, or uncertainty sampling [50, 51], which picks instances with high uncertainty, leverage UQ to guide the AL process effectively. This incorporation of uncertainty awareness facilitates a more intelligent data selection strategy and contributes to building more robust and reliable models.

3 literature review

The field of UQ in OD is an active area of research, and several state-of-the-art methods have emerged. This section describes some of the most notable ones relevant to this thesis.

Recent attention has been directed towards incorporating uncertainty and probabilistic methods within NNs. A predominant focus in this area revolves around adopting Bayesian formalism [52], a framework wherein a prior distribution is defined upon the parameters of a NN. Subsequently, the posterior distribution over these parameters is computed upon exposure to training data. This calculated posterior distribution serves as a pivotal element in quantifying predictive uncertainty. This Bayesian approach allows for a more nuanced and probabilistic perspective in understanding the inherent uncertainties within neural network predictions. By establishing a prior distribution and iteratively updating it based on observed data, researchers and practitioners can derive a more comprehensive understanding of the uncertainty associated with neural network outcomes, providing valuable insights for decision-making and risk assessment in various applications.

3.1 Bayesian Neural Networks (BNNs)

BNNs [53, 54] have gained traction in certain fields. However, they are still not as widely used in practice as their deterministic counterparts, such as traditional NNs trained with methods like back-propagation. Bayesian methods often involve complex computations, especially with high-dimensional data or large-scale models. They also struggle with scalability, particularly in deep learning scenarios. As the model's complexity and the datasets' sizes increase, the computational demands of Bayesian inference can become problematic. In [55], the authors introduce GPBNN. This method combines Gaussian process (GP) regression and Bayesian neural network (BNN) to build surrogate modeling of a computer code output in a hierarchical multi-fidelity context. The authors further quantify the uncertainty of the surrogate model by characterizing the models of different fidelities and their interactions. Their experiments show that the GPBNN method can process noisy and real-life problems. in [56], the authors present a method based on Bayesian inference to incorporate uncertainty into network analyses. The reliability of their method was tested by capturing the local and global properties of simulated networks. Their work shows that Bayesian inference can provide useful information about the underlying uncertainty of a given network

3.2 MC-Dropout

Monte Carlo dropout (MC-dropout), an approximation of Bayesian inference, was introduced by Gal and Ghahramani [38]. MC-dropout offers a method for estimating predictive uncertainty by employing dropout layers [57] during test time. Several studies have explored an approximate Bayesian interpretation of dropout[58, 59]. The widespread adoption of MC-dropout in practical applications can be attributed to its relative simplicity in implementation. However, it is noteworthy that MC-dropout necessitates multiple forward passes during inference that cannot be parallelized, introducing a computational overhead that may be considered a drawback in particular scenarios. Despite this limitation, its popularity persists due to its effectiveness in providing valuable insights into predictive uncertainty within neural network models.

3.3 Deep Ensembles

The authors in [60] proposed using deep ensembles as an uncertainty estimation technique. The authors built upon this foundation by introducing a novel method for training probabilistic NNs that models predictive distributions using a proper scoring rule as the training criterion. Further research has also studied deep ensembles as uncertainty estimation techniques [61, 36, 62]. The fundamental idea of deep ensembles is to train several instances of the same neural network design separately on various random subsets of the training data. These independently trained models' predictions are aggregated to produce a more trustworthy and robust approximation. This method uses the ensemble's diversity of models, which emerges organically from the randomness injected during training. Ensemble members frequently represent the inherent uncertainty in the data in various ways and capture different parts. By aggregating these independent predictions, deep ensembles provide an even more detailed and nuanced overview of uncertainty. However, while ensemble methods, including deep ensembles, offer significant advantages in improving predictive performance and uncertainty estimation, they are not without their downsides. One notable drawback is the increased computational cost of training and maintaining multiple models. Ensembles require training and storing several individual models, leading to higher resource requirements for both time and memory. This can be a limiting factor in applications where computational efficiency is crucial.

3.4 Single shot estimation

Research also contains studies on strategies that use single, deterministic NNs like mean-variance estimation (MVE) [63, 64], Gaussian mixture models (GMM) [65, 66], and deep evidential regression [67, 68, 69]. GMMs provide a probabilistic framework by representing data as a mixture of Gaussian distributions, allowing for capturing complex multi-modal uncertainty. MVE, on the other hand, relies on simple statistics, providing a straightforward approach by estimating the mean and variance of a predictive distribution. This method is computationally efficient but may oversimplify uncertainty representation. Deep evidential regression uses evidence theory to extend NNs to model predictive distributions and their uncertainties. It allows for capturing aleatoric and epistemic uncertainties separately, offering a more nuanced and interpretable uncertainty estimate. While GMMs excel in capturing complex uncertainty structures, MVE emphasizes simplicity and deep evidential regression aims to strike a balance by leveraging the expressive power of DL while maintaining a clear uncertainty modeling framework.

3.5 Augmentation-based estimation

Furthermore, research has attempted to use data augmentations to estimate model uncertainty. Data augmentation is applying modifications to a small dataset to create fresh samples for training, which was first proposed for use in training deep neural networks [18]. The authors in [70] propose a simple but effective method using traditional data augmentation methods such as geometric and color transformations at test time for the estimation of Heteroscedastic Aleatoric Uncertainty. Furthermore, their work showcases the impact of their method via the well-known collection of fundus images obtained from a previous Kaggle competition. Their approach amounts to generating different augmented examples per test case and feeding them to a NN to get a predictive distribution, which can then be used to evaluate the predictive uncertainty. They used data augmentation techniques: random crop and resize, random brightness, hue, saturation, and contrast adjustments, random horizontal and vertical flips, and random rotation. In addition, the work in [71] shows that TTAs can be used to estimate the uncertainty of medical image segmentation. The authors use TTAs and test-time dropout (TTD) to estimate their semantic segmentation model's aleatoric and epistemic uncertainties. The augmentations used in their approach are blurring, down-sampling, spatial transformation, elastic deformations, and system noise. In addition, the authors of [72] have proposed On-Manifold Adversarial Data Augmentation or OMADA, which aims to produce the

most challenging cases by adopting an adversarial attack path that is on-manifold in the latent space of a generative model based on autoencoders and that roughly approximates the decision boundaries between two or more classes. The base idea of OMADA is constructing actual but ambiguous samples for augmentation and input-dependent soft labels for improving the calibration of models.

3.6 Object Detection Uncertainty Estimation

In addition to uncertainty estimation techniques for segmentation and classification networks, the research has abundant uncertainty estimation techniques for OD networks. In such cases, uncertainty lies not only in the detections and their classifications but also in their locations. The work done in [73] provides a workable method for estimating uncertainty in a one-stage object detector while enhancing the baseline approach's detection capabilities. A sizable automobile pedestrian dataset is used to assess the suggested model. The authors also compare aleatoric uncertainties for occluded and fully visible pedestrians. Furthermore, [74] proposes a sampling-free uncertainty estimation method for OD. Based on the authors' claims, it is the first to provide separate uncertainties for each objectness, class, location, and size signal. The authors propose an uncertainty-aware heatmap, exploiting the neighboring bounding boxes the detector provides at inference. In addition, they present a new metric to evaluate location and size uncertainties. The authors of [75] proposed a generative model to estimate box label uncertainties from LiDAR point clouds. Furthermore, they define a new representation of the probabilistic bounding box through spatial distribution. Other work done by the authors of [76] provides a post-processing technique that generates quality and predictive uncertainty estimates for any given NN. A post-processing model fed a structured dataset with a manually crafted set of transparent metrics learns these estimates. The authors discriminate between true and false positives, using the term meta-classification. They further predict IoU values directly, labeling it meta-regression. The meta-classification model's probabilities provide a modeled estimate of predictive uncertainty by attempting to understand the likelihood of success and failure—however, meta-regression results in a quality estimate.

3.7 Aim

To our knowledge, using TTA for OD class and localization uncertainty remains unexplored. Furthermore, while there is no lack of UQ methods in the literature, a need still exists for more robust methods considering real-time systems' memory and time constraints. The work done in this thesis aims to attempt the use of TTAs to quantify the uncertainty of OD models. The proposed method is compared against two methods, ensemble and mc-dropout. In addition, the work in this thesis shows the classification uncertainty of OD models and the spatial uncertainty of detected bounding boxes. Lastly, the proposed method is also used in an AL experiment to show its effect.

4 Methodology

In this chapter we detail the systematic approach utilized to address the hypothesis. We explain the procedures taken to allow for the reproducibility of the study. The further sections explain each part of the approach.

4.1 Model Selection and Dataset

For the work done in this thesis, we chose YOLOv5 as the framework for the methodologies outlined. YOLOv5 has been chosen due to its notable attributes of speed, accuracy, and user-friendly design. Developed to excel in various OD challenges, YOLOv5 emerged as an optimal choice, given its capacity to balance speed and accuracy. This framework has demonstrated its effectiveness across diverse applications, making it a versatile tool for addressing OD problems. Its efficiency and precision align with the objectives of this thesis.

Furthermore, as the use case for the work proposed in this thesis is for AVs, We chose to use the Mapillary dataset [77]. The dataset contains 401 classes. In addition to the bounding boxes and the matching traffic sign templates, the annotators were asked to provide additional attributes for each sign: occluded if the sign is partially occluded; ambiguous if the sign is not classifiable at all (e.g., too small, bad quality, heavy occlusion, etc.); dummy if it looks like a sign but is not (e.g., car stickers, reflections, etc.); out-of-frame if the image border cuts off the sign; included if the sign is part of another bigger sign; and exterior if the sign includes other signs. Some attributes were assigned during localization (if context information is needed). Furthermore, the dataset labels are classified as `other-sign` or have the following shape: `prefix--sign--suffix`. Prefix is one of 4 options, `information, regulatory, complementary, warning` Suffix is one of 9 options `g1, g2, ..., g9`. Excessive labels in a machine learning model can introduce several challenges and potential drawbacks. First and foremost, a surplus of labels can lead to overfitting [78], where the model becomes overly tailored to the training data and loses its ability to generalize well to unseen instances [79]. Overfitting occurs when the model memorizes the training set instead of learning the underlying patterns, hindering its performance on new and diverse data. Additionally, an abundance of labels increases the dimensionality of the learning task, making the model more computationally demanding and resource-intensive. This escalates training times and leads to difficulties in model interpretation and scalability. Furthermore, managing many labels requires extensive uniformly labeled data, which may not always be readily available and can be costly to acquire. For this reason, we chose to group labels into a smaller number of super-classes. Furthermore, it's believed that grouping boxes of different classes under the same label will increase the models' uncertainty, which the work of this thesis aims to quantify.

A further understanding of the labels was needed to understand which grouping to use. The coming section shows examples of each label to explain each part of the label. Labels are grouped using regular expressions (regexes), and a sample of six images is used for each regex. The regex shows on top of each of the grouped images.

Figure 4.1 Examples of combining different suffixes and prefixes for the dataset [77]

Figure 4.1 shows the effect of changing the prefixes and suffixes of different boxes. `.*` disregards the type of sign, focusing only on the prefix and suffix. Furthermore, the label on top of each box can be seen. This shows a clear variance in the images of the same prefix-suffix combination.

Next, a study focusing on the suffixes was conducted to understand whether they can be completely dropped. Using the same regex idea as before, a combination of all regulatory–keep-right signs have been picked with different suffixes to show the differences.

Figure 4.2 Examples of comparing images with the same label but different suffixes

Figure 4.2 shows that some boxes within the same label and suffix also show some variance. Next, an analysis of the label frequency of each box was also conducted. The goal was to show the distribution of classes across boxes to see if combinations can be more uniform than others. First, it was observed that the class `other-sign` exists with around $120K$ occurrences, and every other class has at most 2500 occurrences. For such reason, the frequency of the class `other-sign` has been dropped to not skew the y-axis of the graph.

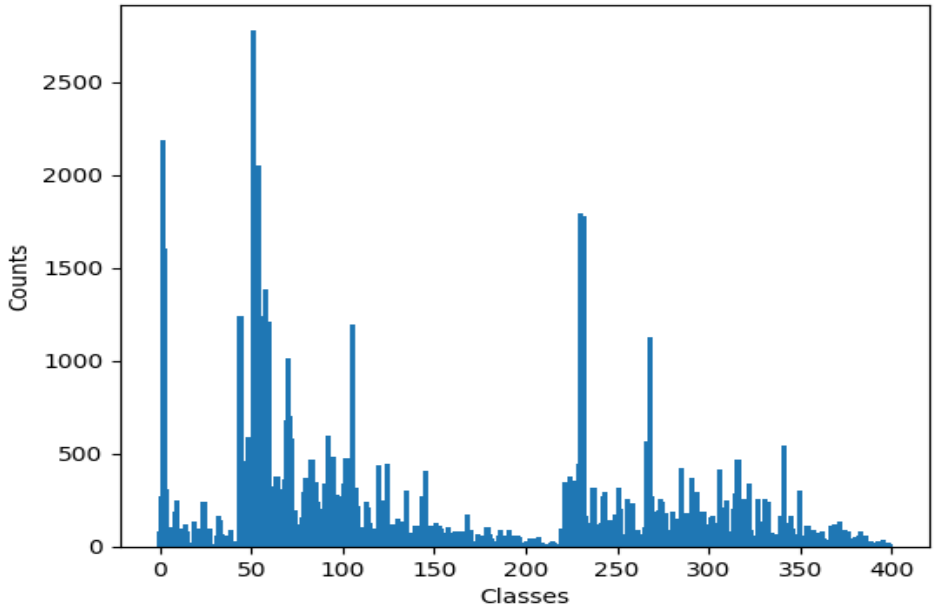


Figure 4.3 Mapillary's label frequency for 400 classes.

Figure 4.3 shows the distribution of labels across boxes when no change has been made to the classes and no grouping. There's a clear skew in the distribution. One possible grouping of the previous 400 classes would be to drop all suffixes and prefixes and group all boxes with the same sign as one class. This results in 235 classes. The distribution of such a combination can be seen in Figure 4.4a. However, 235 classes is still a large number. Another option is to combine similar classes; for example, `maximum_speed_limit{V}` for `V` in

$$10, 20, 30...., 100$$

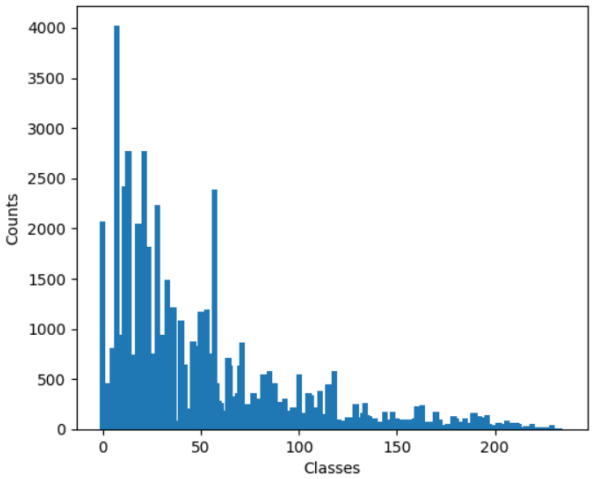
could be combined into one class detailing a speed limit of any kind. In addition, `one-way-{Dir}` for `Dir` in

$$left, right, straight$$

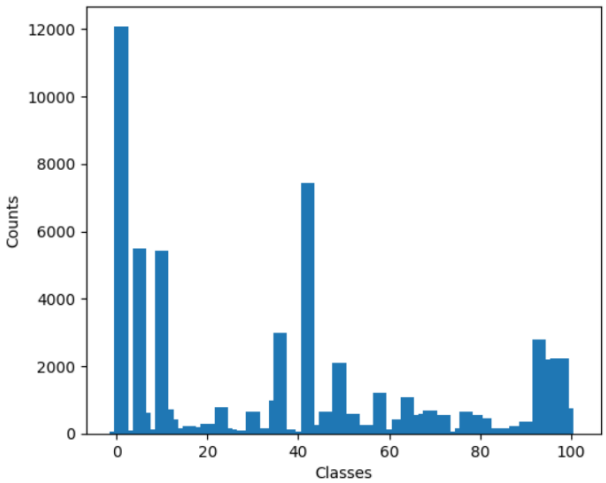
could be similarly combined into a class detailing `one-way`. Combining all classes with similar features, as explained, produces a distribution with 100 classes seen in Figure 4.4b. Furthermore, combining classes that could be under the same name, for example, `speed limit`, `weight limit`, and `width limit` could all be under the name `limit`. The number of classes can then be dropped to 75 classes with distribution seen in Figure 4.4c. One might argue that this produces combinations that are not necessarily groupable. Furthermore, the distribution is still skewed and contains many classes. As such, a more trivial grouping could also be proposed using only the labels' 4 prefixes

$$Information, Regulatory, Complementary, Warning$$

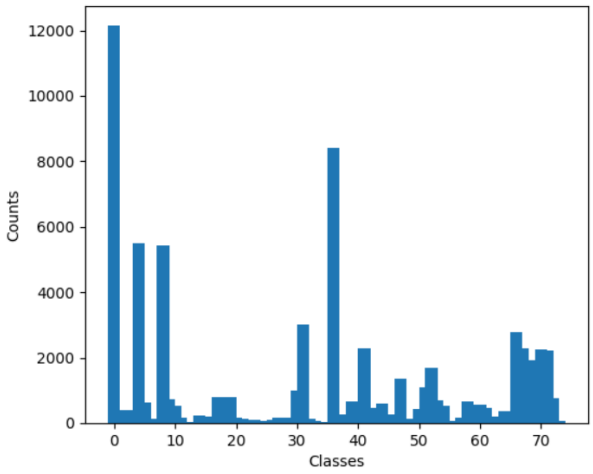
offering the distribution seen in Figure 4.4d. Although still somewhat skewed, it offers the fewest classes, and the class grouping offers an inherent uncertainty in class predictions.



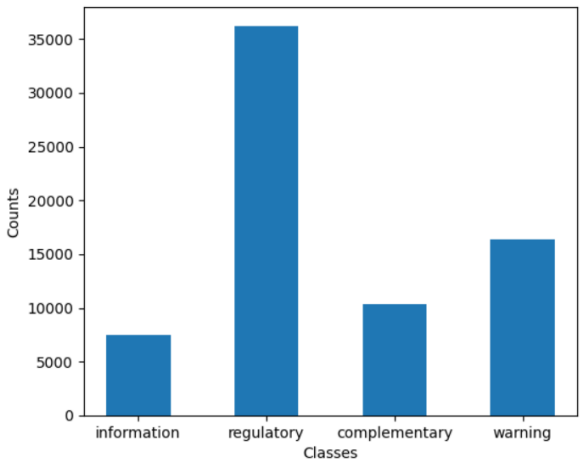
(a) 235 Classes



(b) 100 Classes



(c) 75 Classes

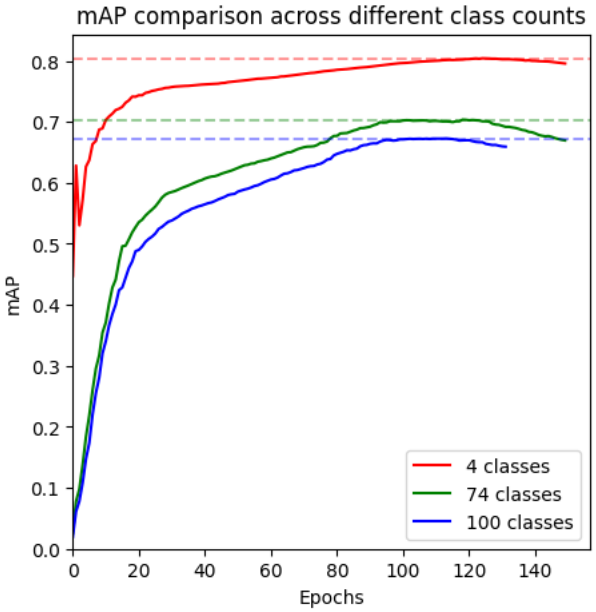


(d) 4 Classes

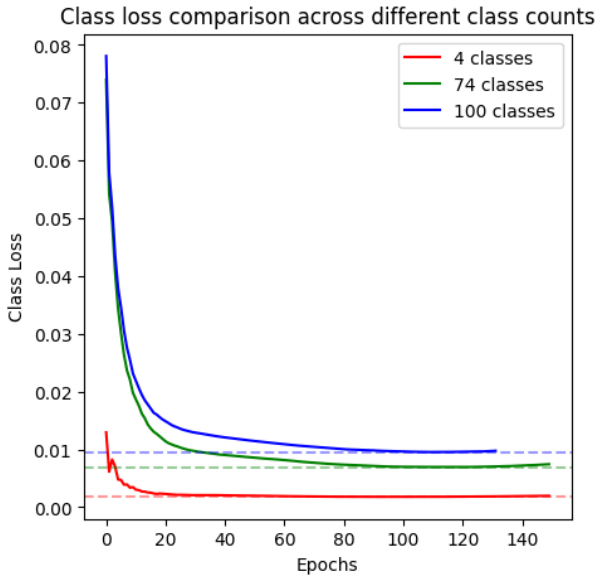
Figure 4.4 Label distribution across boxes for different class groupings

To further decide which distribution to use, models trained on each distribution mentioned were compared to see which offers the best performance. First, all models had similar box losses, showing they could all detect the boxes' locations. Models' mAPs and class losses were metrics to decide which model excels in performance. Mean Average Precision (mAP) [80, 81] is a metric utilized to evaluate OD models, representing a standard performance measure. It assesses the accuracy and robustness of a model in identifying and localizing objects within images across various classes. mAP is the mean of each class's average precision (AP) scores. AP, in turn, quantifies the precision-recall trade-off for a specific class, reflecting the model's ability to precisely identify instances of that class while also ensuring high recall.

All models had the same architecture (YOLOv5m) and the same default parameters. All models were trained for 250 epochs with early stopping enabled with patience of 20 epochs. An image size of 1280 was used with a batch size of 64. Figure 4.5a shows the mAP graphs of models trained on each distribution, while Figure 4.5b shows the class loss graphs. It's worth noting that although the 401 classes graph is not shown, it's trivial to assume it will follow the same pattern. This shows that whenever the number of classes goes up, there is a decrease in model performance, which was expected. As such, the decision to use the 4-class distribution was made.



(a) mAP Comparison of different distributions



(b) Loss Comparison of different distributions

4.2 Training

After selecting the dataset along with its label distribution, further inspection of the dataset was conducted. The dataset contained 36000 training images, 5300 validation images, and 10500 test images. However, we observed that the test dataset had no labels. As such, the testing dataset could be used for uncertainty estimates since labels were not needed but could not be used to assess model performance. In addition, we needed a select set of images that were not used in training but could be used for oracle selection during AL experiments. Hence, we divided the training dataset into Train-Select-Test splits.

We analyzed the model's performance on different training data sizes to determine the optimal size of the training dataset needed. The analyses used varying training set percentages. The experimentation began with 10 percent of the dataset allocated for training, serving as an initial baseline. Subsequently, the training set size was exponentially increased, and the model was trained and evaluated at each increment. The metric used for evaluation was the mAP score. This process was to find an 'elbow' in the mAP score, signifying the optimal training set size. We aimed to balance getting a well-trained initial model and leaving room for improvement during AL iterations. This allowed us to ensure efficient utilization of the available

dataset while maintaining model generalization across subsets. Figure 4.6 shows the mAP of models trained on different percentages of the training data. The X-axis is increasing in exponential order. The figure shows an elbow on the 19-22% and a substantial increase at 40%. Aiming for a larger size for the select dataset, the choice was made to use 22 percent as the training size and 15 percent for the test set, allowing for a 63 percent select set.

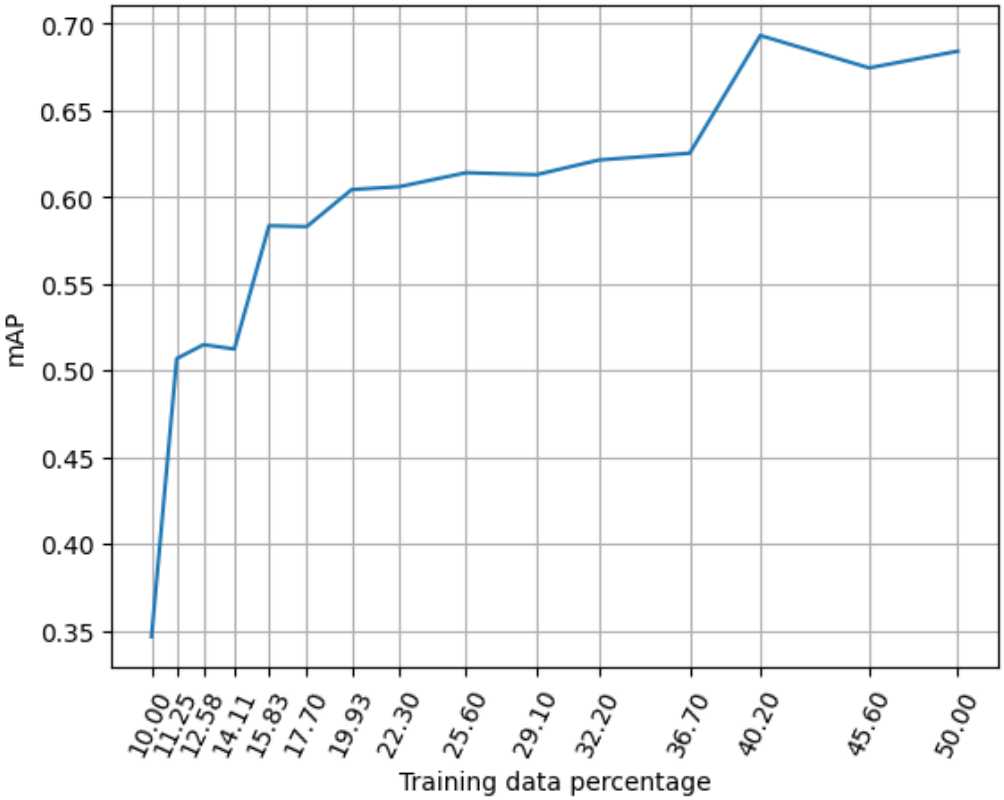


Figure 4.6 Model mAP through different training data percentages

We trained ten models on the new dataset split to create an ensemble of YOLO models for uncertainty calculations. Each model had different initial weights to ensure differences in prediction. Figure 4.7 shows the mAP graphs of the models

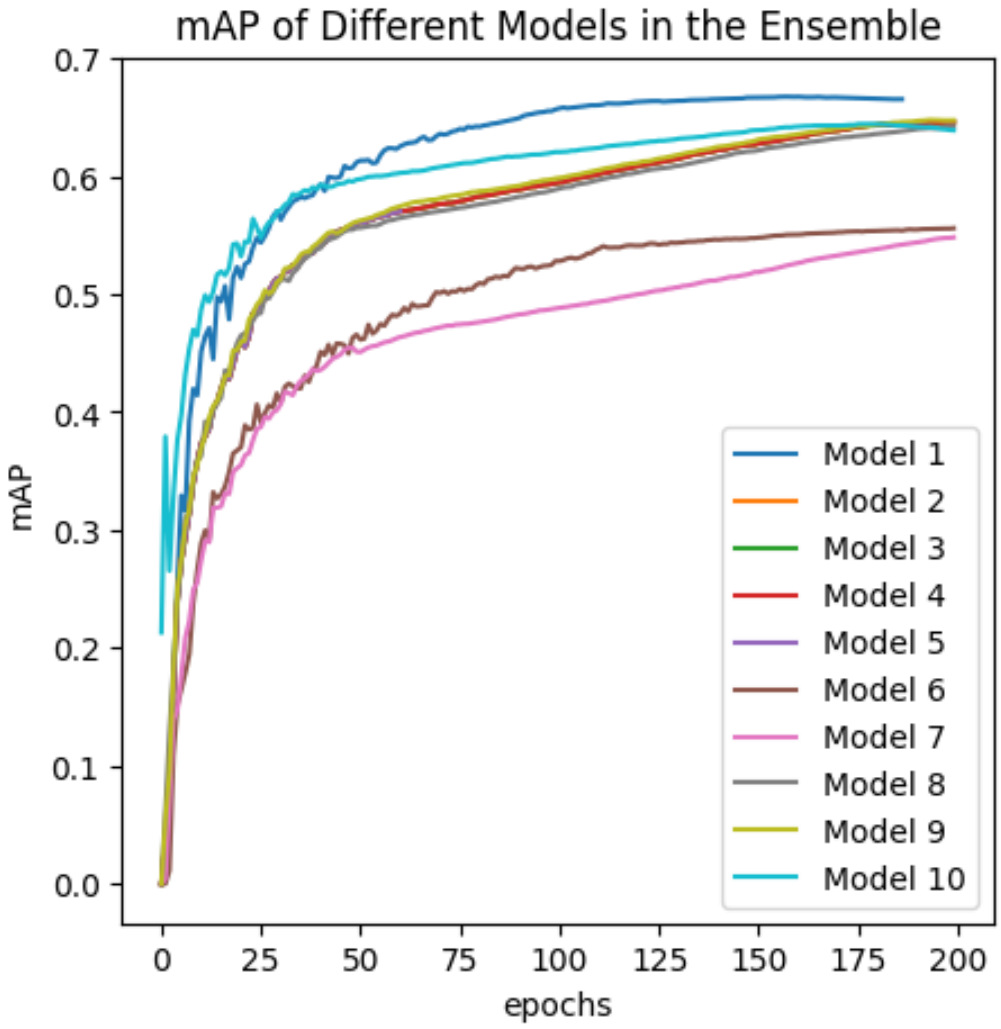


Figure 4.7 mAP on trainset through training epochs comparison of models in the Ensemble

4.3 Augmentations

We conducted experiments to show the best augmentation options for the model. We aimed to show the result of each option on the model outputs. Based on previous research, the augmentation options focused on for this research are the following

- Image Gaussian blurring: A random filter size is selected between 5 and 30, with a sigma of (0.1, 5)
- Random value changing: Adding/subtracting a random value to 30 percent of the image pixels.
- Colour jitters: A color jitter with a brightness of 0.5 and a hue of 0.3.
- Elastic transform: Random elastic transform with a random alpha between 50 and 200.
- Rotation: Random image rotation between -20 to 20 degrees.
- Polarization: posterize the image with bits value between 1 and 4.
- Salt & pepper: adds salt and pepper noise to 30 percent of the pixels.
- Random Noise: Random noise with a variance of 0,9.

Other augmentations with no parameters were used, such as:

- Gaussian noise.
- Poisson noise.
- Contrast enhancing.
- Gamma correction.
- Horizontal flipping.
- Vertical flipping.
- Equalization.

Apart from rotation and flipping, all augmentations keep pixels in the same locations. While flipping has trivial reverse functions, rotations aren't as trivial; as the rotation angle increases, loss of information around the corners occurs. Although it is possible to pad the image to reduce the loss of information, changing the image dimensions and aspect ratio poses an issue. Therefore, the choice to drop random rotations was made.

For the remaining choices, we chose to observe the percentage of false negatives added to the model prediction due to the addition of the augmentation. A random sample 500 images was deliberately selected from the dataset for each augmentation iteration. All augmentations were tested on the same sample. Subsequently, the model underwent inference with two conditions: once without augmenting the image and once after applying the selected augmentation. The objective was to assess the impact of each augmentation on the model's performance. Specifically, a count was maintained for each predicted bounding box generated by the model. This count included instances where the bounding boxes were successfully detected and instances where they remained undetected following the augmentation application. This only checked that a box that the model found before the augmentation was applied, regardless of whether the initial box was present in the ground truth or not. This analysis aimed to quantify the influence of each augmentation on the model's ability to accurately perceive and delineate object boundaries, providing valuable insights into the augmentation techniques' efficacy. Figure 4.8 shows each augmentation's observed effects.

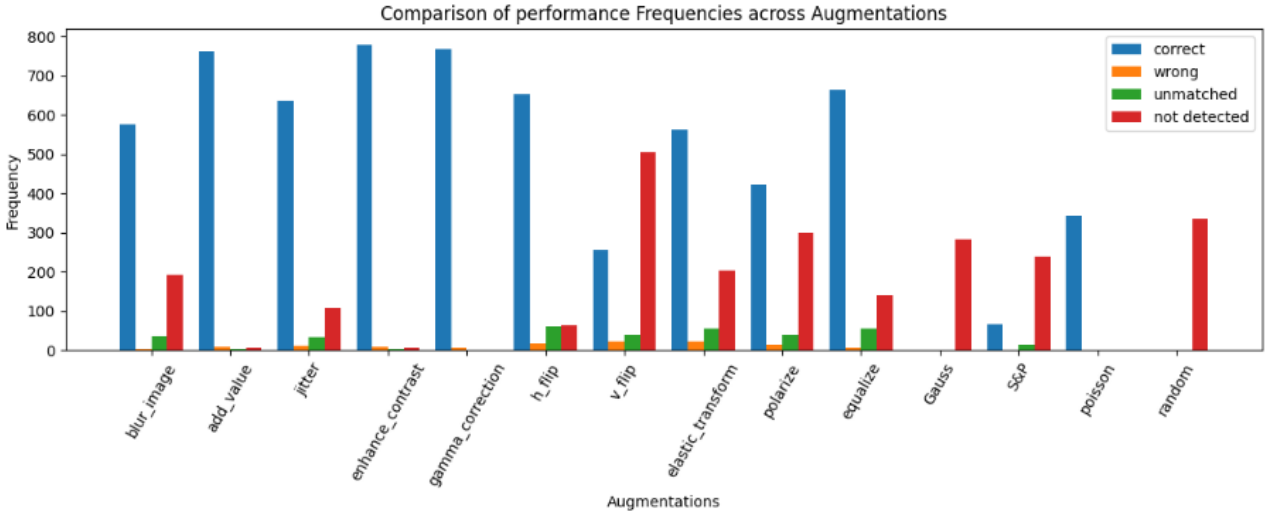


Figure 4.8 Effect of Augmentations on model performance

We applied a criterion to establish correspondences between bounding boxes, using an Intersection over union (IoU) threshold set at 70%. This threshold determined whether two boxes were considered overlapping. Subsequently, if the overlapping boxes exhibited the same class, they were labeled as "correct";

conversely, if they belonged to different classes, the match was deemed "wrong". Furthermore, in cases where a bounding box remained unmatched with any other box post-augmentation (false-negative), it was categorized as "not detected". Additionally, instances where a box emerged following augmentation yet did not have a corresponding counterpart in the pre-augmentation phase (false-positive) were counted as "unmatched." This process aimed to comprehensively evaluate the matching process, providing a detailed account of correct matches, incorrect matches, false-negative scenarios, and false-positive occurrences resulting from the augmentation process. An aggregation of the values over the total number of boxes was used to observe only false negatives to show the distribution in Figure 4.9. We chose a threshold of 25 percent to keep some augmentations with a higher error rate increasing uncertainty, but not those showing extreme error rates. The red dashed line denotes the threshold where all augmentation below or exactly on the line would be considered, and those above would be dropped.

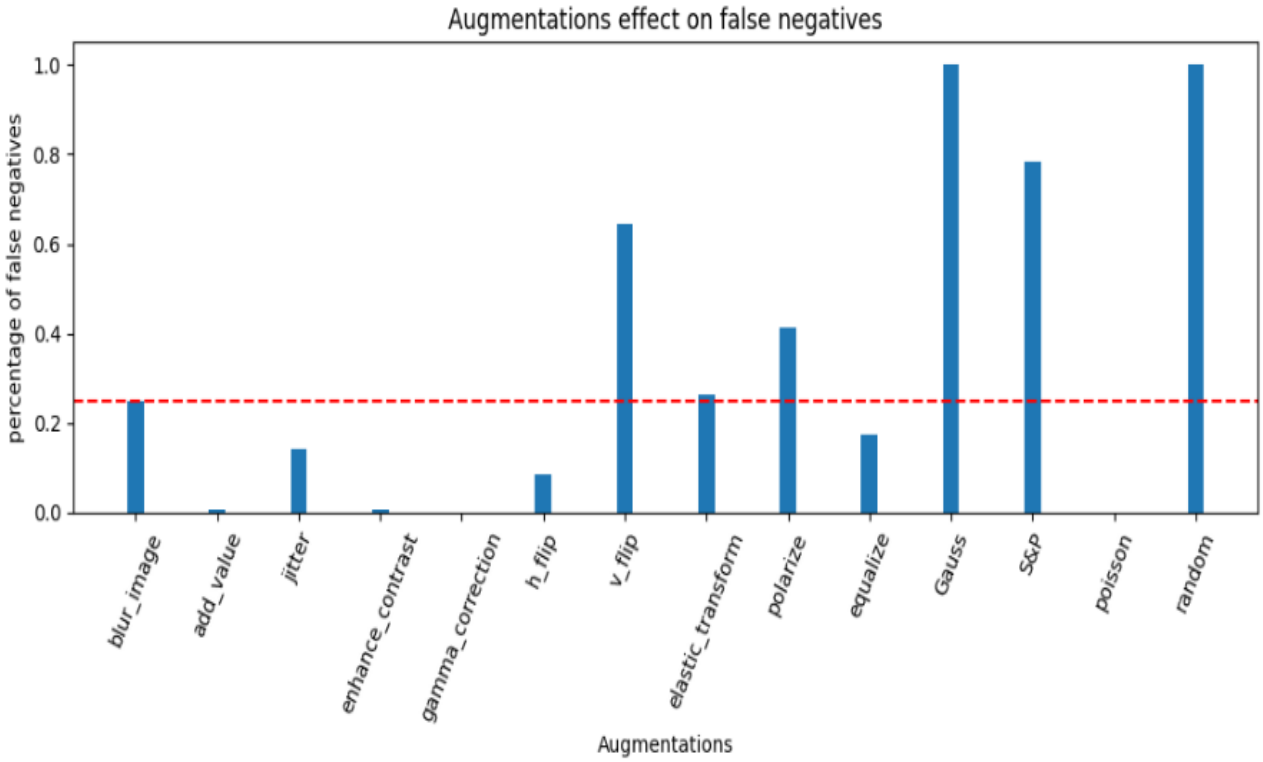


Figure 4.9 Percentage of false negatives added by augmentations

The augmentations kept after the experiment are the following

- Image blurring
- Random value changing
- Colour jitters
- Contrast enhancing
- Gamma correction
- Horizontal flipping
- Elastic transform
- Equalization
- Poisson noise

4.4 Inference

In this section, the inference of each method is explained showing how outputs were created for each of them for the uncertainty calculations. The focus remains on Ensemble methods, MC-dropout, and the proposed TTA approach. Each of the following sections focuses on one approach individually.

4.4.1 Ensemble

For ensemble method inference, the process is straightforward. As previously detailed, ten models were trained, each with slight differences in weight initialization to induce variances among the individual models. The ensemble inference involves aggregating predictions from all models for a more reliable prediction. This approach utilizes the diversity of the models within the ensemble, providing an estimation of uncertainty.

Ensembles can be seen as approximate Bayesian models. The aggregation of predictions from multiple models in an ensemble resembles averaging over different hypotheses, which aligns with the principles of Bayesian inference. The combination of models allows for capturing different sources of uncertainty, akin to incorporating prior beliefs in Bayesian statistics. The next equation adapted from [60] shows a representation of how ensemble methods relate to Bayesian inference

$$P(y|D) = \frac{1}{M} \sum_{m=1}^{M} P\phi_m(y|D, \phi_m) \tag{4.1}$$

where:

- M: The number of models in the ensemble
- D: The datapoint used for inference
- ϕ_m: Refers to the weights of model m

This equation illustrates how ensemble methods effectively incorporate Bayesian principles by considering the likelihood of predictions from each model and their associated posterior probabilities.

Using ensemble inference, ten outputs are generated for each input image, each slightly different.

4.4.2 MC-Dropout

For MC-Dropout inference, the methodology involves incorporating dropout during inference. Predictions are obtained by running the model multiple times with dropout activated and aggregating the results. This dropout-based approach estimates the model uncertainty and can be seen as a form of approximate Bayesian inference. Utilizing the equation from the previous section, we sum over a number of iterations instead of summing over models. The MC-Dropout prediction $P(y|D)$ can be expressed as:

$$P(y|D) = \frac{1}{I} \sum_{i=1}^{I} P(y|D, \phi_{\text{dropout activated}}) \tag{4.2}$$

Where:

- I: The number of stochastic forward passes.
- D: The datapoint used for inference
- ϕ: Refers to the weights of model

The original YOLOv5 codebase lacked built-in support for incorporating dropout in any convolutional layer, necessitating modifications for this requirement. Consequently, we edited the codebase to add a dropout layer to the final convolutional layer. We used a dropout rate of 50 percent, with the dropout layer remaining active during the inference stage. However, we observed that one dropout layer doesn't add enough changes between each subsequent prediction; hence, a dropout layer of the rate of $.02$ was added to every other convolutional block. This ensured a subtle change between the predictions of each forward pass.

Following the research, we chose 50 as the number of forward passes for each inference step, generating 50 outputs for each input image, each different than the other.

4.4.3 Test-Time Augmentations (TTA)

For each TTA inference step, several generated images are generated. The image undergoes three augmentations sequentially. A random subset of 3 augmentations is chosen from the list shown in 4.3. Since we have a finite number of augmentations (9 in total), we define a complex augmentation as a collection of 3 augmentations done in any order. This creates a list of 84 unique augmentations ($9C3$).

To show that using augmentations is an approximation of Bayesian inference, we can utilize the equations in the previous 2 sections. The TTA prediction $P(y|D)$ can be expressed as:

$$P(y|D) = \frac{1}{I}\sum_{i=1}^{I} P(y|A_i(D), \phi) \tag{4.3}$$

Where:

- I: The selected number of images.
- D: Is the datapoint used for inference
- A_i: The complex augmentation chosen from the list
- ϕ: Refers to the weights of model

This equation illustrates how the TTA method effectively incorporates Bayesian principles by considering the likelihood of predictions from each data point and their associated posterior probabilities.

As this is closest to ensemble method inference, ten images were chosen for each input image, allowing for ten different outputs. It's worth noting that, unlike MC-Dropout, this can be done as a vectorized batch prediction. In addition, unlike ensemble methods, it significantly reduces the memory cost as only 1 model needs to be maintained and used on inference.

Ten complex augmentations are chosen randomly for each step, generating ten unique images at inference time. These are then used to generate the ten outputs, each slightly different from the other.

4.5 Uncertainty calculations

After generating n outputs for each inference method, a calculation of the uncertainty of each method for each image is to be calculated. The following subsections explain two possibilities for calculating uncertainties. The first explains a pixel-level uncertainty where each pixel is given an uncertainty value, while the other focuses on a higher-order calculation giving uncertainty to components.

4.5.1 Pixel-level

The pixel level calculation creates a channel for each predicted box. Each pixel in each channel represents the class probability of the box. Most pixels in one channel exist outside of the box; we need a representation of the probabilities of those pixels. Option 1 gives an equal probability to each class, while option

2 adds the notion of background class. For this calculation, we chose the latter. All predicted probability scores are appended with a background probability of 0 percent when the pixel lies inside a box and 100 percent when outside the box. Each pixel in its respective channel is then given the probability scores of its respective model output. The pixel-level entropy of the pixel-level expectation of the probability scores then returns the Total uncertainty of each pixel given all outputs.

$$H(X) = -\sum_{i=1}^{n} \mathbb{E}_p[p_i \cdot \log(p_i)] \tag{4.4}$$

where:

- $H(X)$: Entropy of the input pixel X.
- $\sum_{i=1}^{n}$: Summation over all possible outcomes of X.
- $\mathbb{E}_p$: Expectation with respect to the probability distribution p.
- p_i: Probability of outcome i occurring.
- $\log$: Natural logarithm.

The aleatoric uncertainty can then be given by the pixel level entropy of the pixel level expectation of the probability scores

$$U_a(X) = \mathbb{E}_p \left[-\sum_{i=1}^{n} p_i \cdot \log(p_i) \right] \tag{4.5}$$

- $U_a(X)$: Aleatoric uncertainty of each pixel X.
- $\mathbb{E}_p$: Expectation with respect to the probability distribution p.
- $\sum_{i=1}^{n}$: Summation over all possible outcomes of X.
- p_i: Probability of outcome i occurring.
- $\log$: Natural logarithm.

The epistemic uncertainty can then be given by subtracting the aleatoric uncertainty from the total uncertainty

$$U_e(X) = H(X) - U_a(X) \tag{4.6}$$

This returns an image-like matrix where each pixel now shows the pixel-level uncertainty.

4.5.2 Component-level

The component level uncertainty, on the other hand, focuses only on the components predicted and not on each pixel. Each component is created from an initial prediction treated as an anchor. For TTA, the anchor is created by creating an extra initial inference without any augmentations, whereas for MC-dropout, an extra initial inference is created without dropout. Finally, the model with the best performance (denoted by the best mAP) is used as an anchor generator for the ensemble. It's worth noting that this approach introduces some ambiguity. We assume the anchor predictor is the best-performing prediction. However, a case can be made that the initial anchor prediction was a false positive or a false negative. a box not appearing in the anchor predictor does not create a component. If all models in the ensemble, for example, see a box that the anchor predictor does not see, that box is dropped from the uncertainty calculations.

For each anchor, we group every overlapping box (IoU threshold of 80%). The grouping can be used to create a component for each anchor. The component can be calculated with the union or the intersection of all boxes. The intersection of all boxes can be used to calculate the box class uncertainty, while the union minus the intersection can denote the localization uncertainty. This works since the union contains

pixels in some boxes but not others, which entails that the predictions did not agree on the box's location. Figure 4.10 visualizes the different regions of the component.

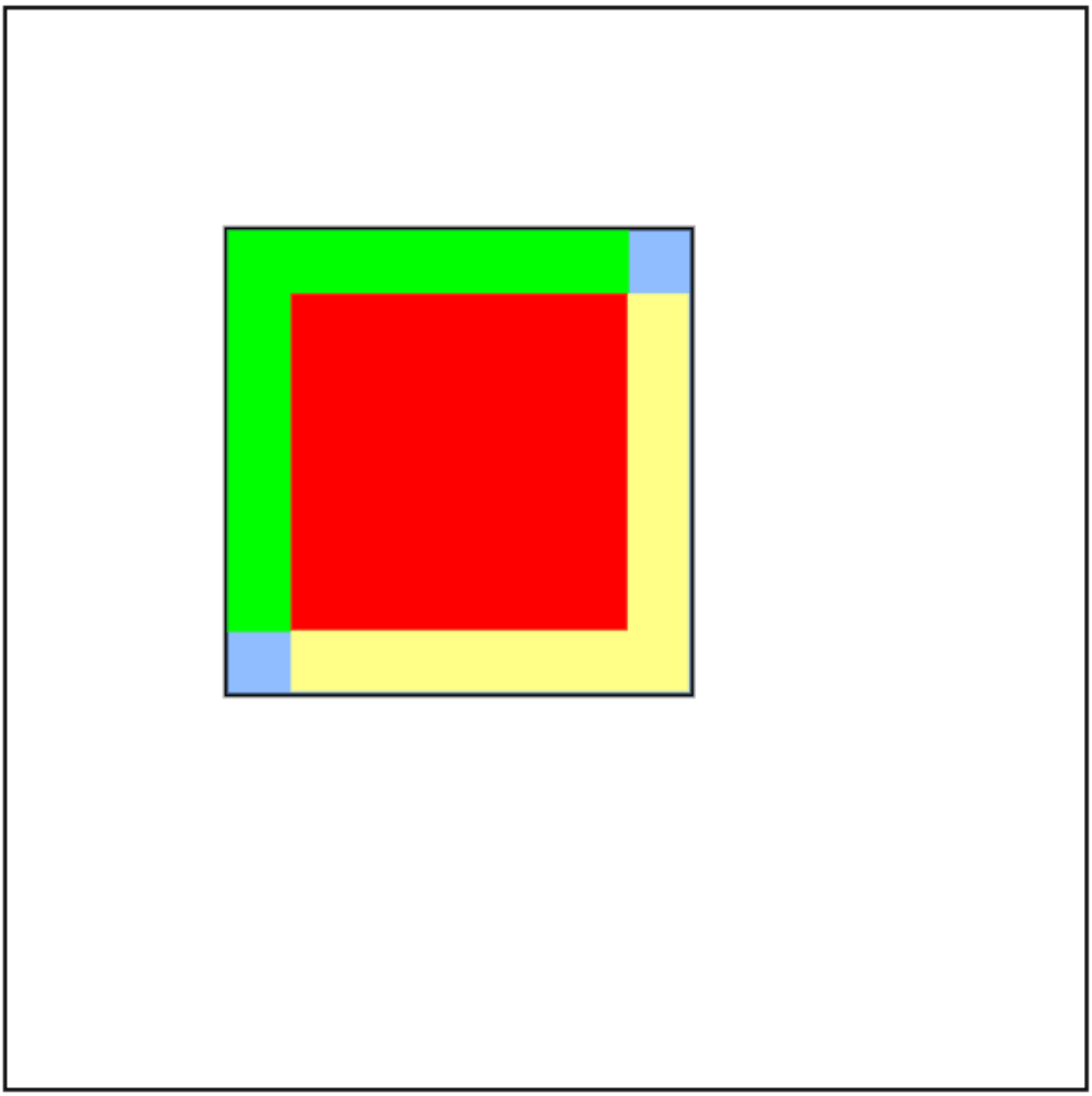


Figure 4.10 Visualization of component level calculation. The white box represents the image, the blue box shows the union of the two overlapping boxes(Green and yellow), and the red box shows the intersection.

Each component is a $[W, H]$ matrix, where W is the component's width and H is the component's height. The calculations are similar to those used for the pixel-level calculation. However, since we assume the entire component is the object, there is no notion of a background object. a 'pixel' not in one of the boxes is denoted by equal probabilities for each class, giving the highest uncertainty. Following the notation in Figure 4.10, assume only two classes; each pixel then has a probability score $[P_1, P_2]$. for the green box, all pixels lying within that box have $[P_{G1}, P_{G2}]$ where all those outside have $[.5, .5]$. Similarly for the yellow box $[P_{Y1}, P_{Y2}]$. The blue corners can then be given the score $[.5, .5]$ since they lie outside of both boxes, while the red can be calculated as the following $[\frac{P_{G1}+P_{Y1}}{2}, \frac{P_{G2}+P_{Y2}}{2}]$. Finally, points lying inside one box but not the other, assume green, can be calculated as $[\frac{P_{G1}+.5}{2}, \frac{P_{G2}+.5}{2}]$. This can be trivially expanded to more boxes with higher class counts.

This shows that the closer the boxes agree on the box location and the closer they are together, the lower the uncertainty values on the edges, thereby entailing lower localization uncertainty. Furthermore, the intersection contains the same value for all pixels, denoting the mean class probability which is used to calculate the class uncertainty.

4.6 Image-level Uncertainty

An aggregation is to be used to create an uncertainty value for each image. The following aggregation criteria have been selected for further analysis

- Max: Trivially uses the maximum value present for the overall uncertainty
- Mean: Calculate the average of each pixel for pixel level or the average of each component
- Weighted Mean: weighted mean where the log size of each box is used as a weight
- Thresholding: Uses a threshold value dropping all boxes surpassing the threshold and then using a trivial average
- Weighted Thresholding: similar to thresholding but utilizes the weighted mean.

Thresholding prevents aggregation from being skewed by smaller objects, pushing the uncertainty higher. These methods offer an image-level uncertainty value that can be used for AL experiments.

4.7 Active Learning

As explained above, AL allows ML models to select which data points to train on next. Uncertainty estimates derived from BNN play a crucial role in assessing the models' confidence in their predictions. By selecting images showing high uncertainty, the process aims to improve the models' performance with fewer labeled examples. This iterative approach involves the model iteratively retraining on the labeled data, continuously updating its understanding of the data distribution. The incorporation of UQ guides the process and provides insights into areas where the model may benefit from additional training.

Within each iteration, a traversal of the select set is undertaken. Employing the chosen UQ method, the algorithm calculates image-level uncertainty estimates about each image within the dataset. Subsequently, the data points with the highest uncertainty scores are incorporated into the training set for subsequent model training. We repeat the cycle to improve the model performance. We chose to include the top 5 percent of the total dataset size in the training set after each iteration. In each iteration, 1500 images with the highest uncertainty are moved to the training set, and then a new model is trained on the new data. Each experiment ran five iterations, comparing random selection, MC-dropout, and the proposed TTA approach.

We created an initial model, namely model 0, trained on the train set. For each subsequent iteration i, we used model $i-1$ to evaluate the uncertainty of the select set's images. After the most uncertain images are added to the train set, model $i-1$ is retrained on the new dataset to create model i. The process is repeated for a total of 5 iterations. This process works for MC-dropout and TTA; however, it is slightly different for deep ensembles. Every iteration must train 10 models for the uncertainty calculation in such cases. This creates 60 trained models for five iterations. 10 initial models and 10 for every iteration.

5 Experimental results

5.1 Experimental setup

This chapter shows and explains the results of the aforementioned approach. First, we visualize the uncertainty on images of the test set. We use the Mapillary unlabeled test set for this since labels are not needed for the uncertainty visualizations. Furthermore, since the data weren't used for training, it also showcases the results of the aforementioned method on unseen images.

Next, we statistically analyze the uncertainty calculations of each method (Ensembles, MC-dropout, and TTA) on the validation dataset; this aims to show the distribution of image and component-level uncertainties. Which we use to choose a threshold value for the thresholding and weighted thresholding uncertainty methods.

After, we observe the similarity of the available methods using different uncertainty aggregation criteria.

Finally, we show the results of the AL experiments. As explained above, deep ensemble methods require creating and training too many currently unfeasible model counts. For such reasoning, deep ensembles were not used in the AL experiments. A random selection criteria is used and compared with both TTA and MC-dropout to use as a baseline for improvements.

5.2 Uncertainty visualizations

In this section, we visualize the pixel and component-level uncertainty on unseen images.

5.2.1 Pixel-Level

Figure 5.1 shows a sample of the results of the pixel-level uncertainty estimates for each method. ENS stands for deep ensemble methods, MCD for MC-Dropout, and TTA for the proposed test-time augmentations technique. We observe that the uncertainty estimates are low, never reaching 1. Furthermore, the center of the boxes has the highest uncertainty since pixel-level border pixels contain low uncertainty because they're assumed to be background classes. This, however, bounds the uncertainty distribution of the dataset to low values, making it hard to choose a threshold for highly uncertain data points. The distribution of the pixel-level uncertainty on the validation dataset can be seen in the next section 5.3.1. Furthermore, more figures can be seen in the appendix A.1. We calculated the uncertainty on a component level to overcome the downsides of pixel-level calculations.

5.2.2 Component-Level

In component-level calculation, each component is a stand-alone representation; however, for ease of visualization, we have all components in an image on the same figure.

It's worth noting that methods do not need to agree on the number of components available in each image. Figure 5.2 shows the results of the three methods on the same image. In contrast to the pixel-level estimation, the borders of the box contain the highest uncertainties, while the center of each component has a uniform value for the class uncertainty. It can be observed that the edges of the bounding box contain the highest uncertainty, which can be explained as the localization uncertainty of the box, where the center shows a uniform value denoting the class uncertainty. More figures can be seen in the appendix A.2

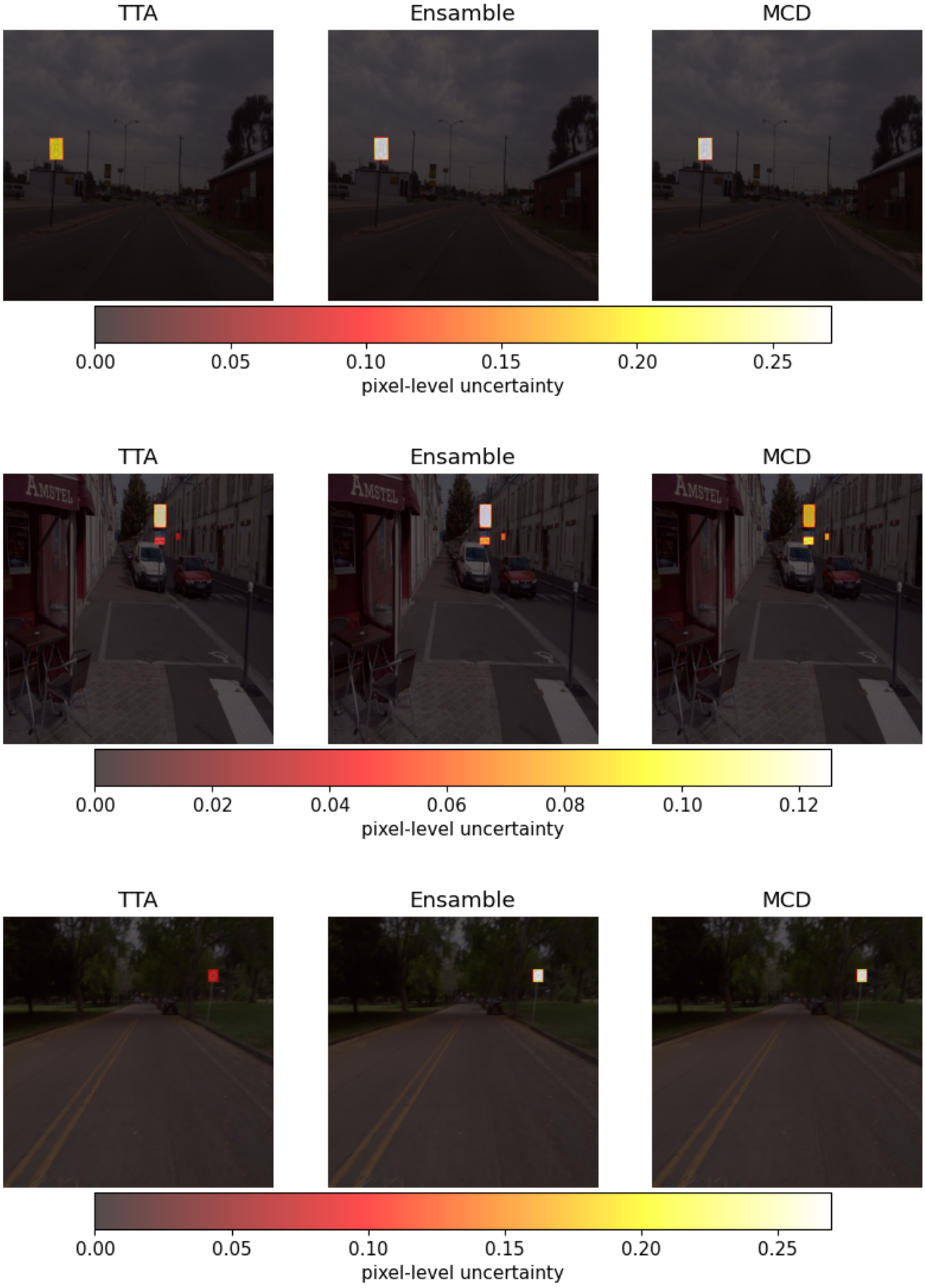


Figure 5.1 Pixel level uncertainty estimation of different methods

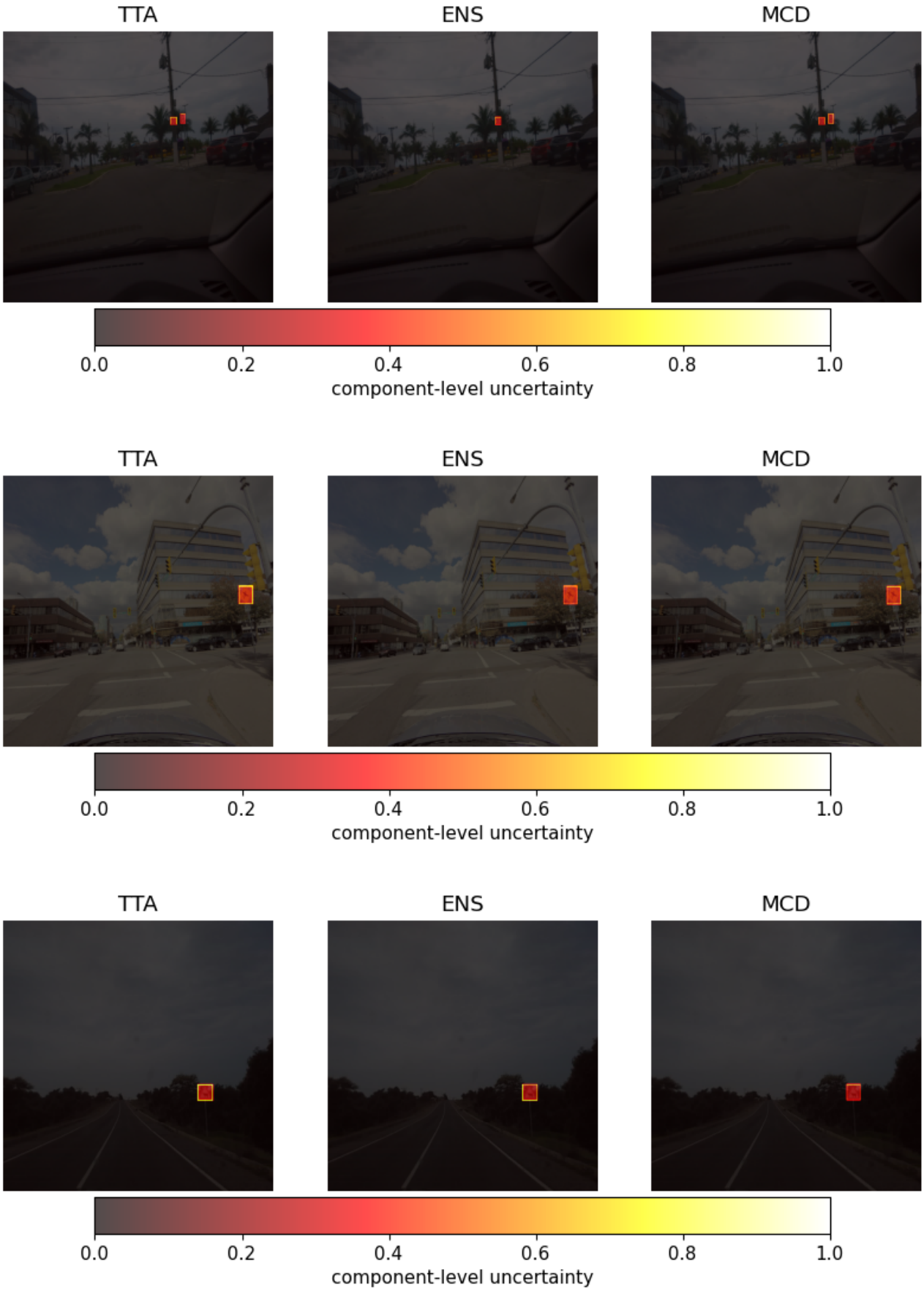


Figure 5.2 Component-level uncertainty estimation of different methods

5.3 Uncertainty Analysis

5.3.1 Pixel-level Uncertainty

For pixel-wise uncertainty, we consider a connected component to be a series of connected uncertain pixels with no gaps. The mean of the connected component's pixels or the connected component's max uncertainty can then be used to estimate the entire component's uncertainty. In the following section, the pixel-level uncertainty distributions are shown. The y-axis in the plots is log-scaled, and the uncertainty values have been normalized between 0 and 1 for ease of visualization.

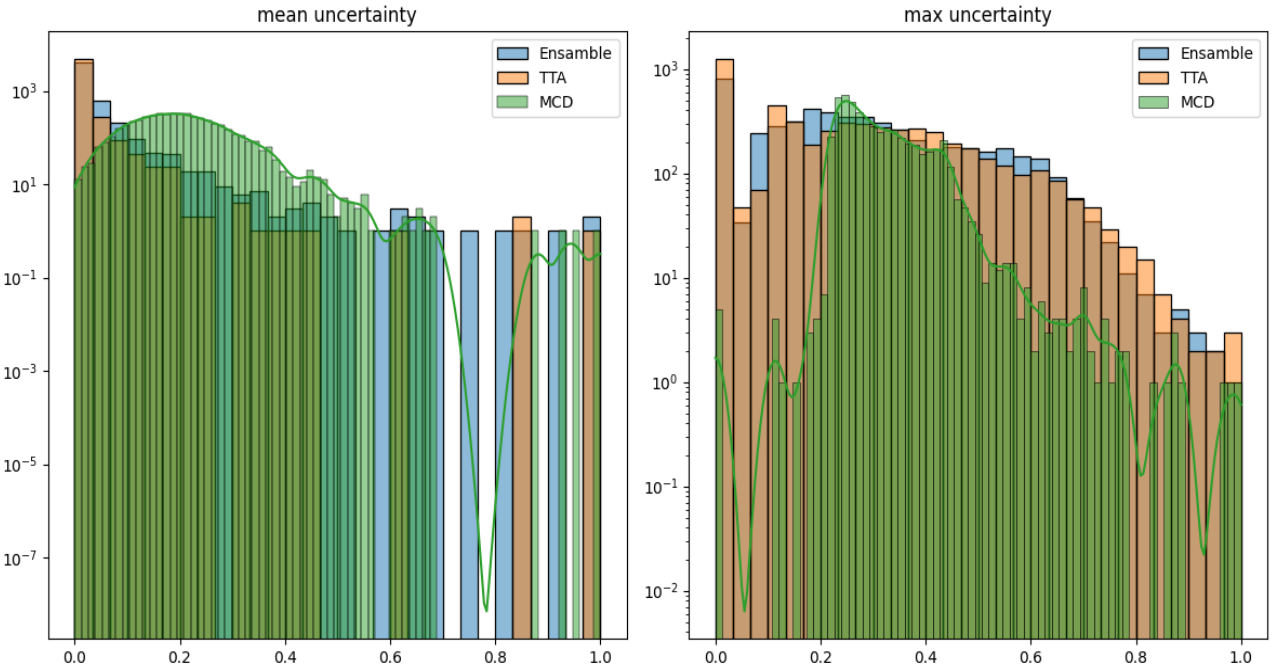


Figure 5.3 Pixel-level uncertainty distribution across data samples

Figure 5.3 shows the distributions of the pixel-wise uncertainties. The graph shows similar distributions between ensembles and TTA but a different one for MC-dropout.

5.3.2 Component-level Uncertainty

Since class uncertainties are calculated using the intersection of overlapping boxes, the entire component has the same uncertainty value. In the following section, the component-level uncertainty distributions are shown.

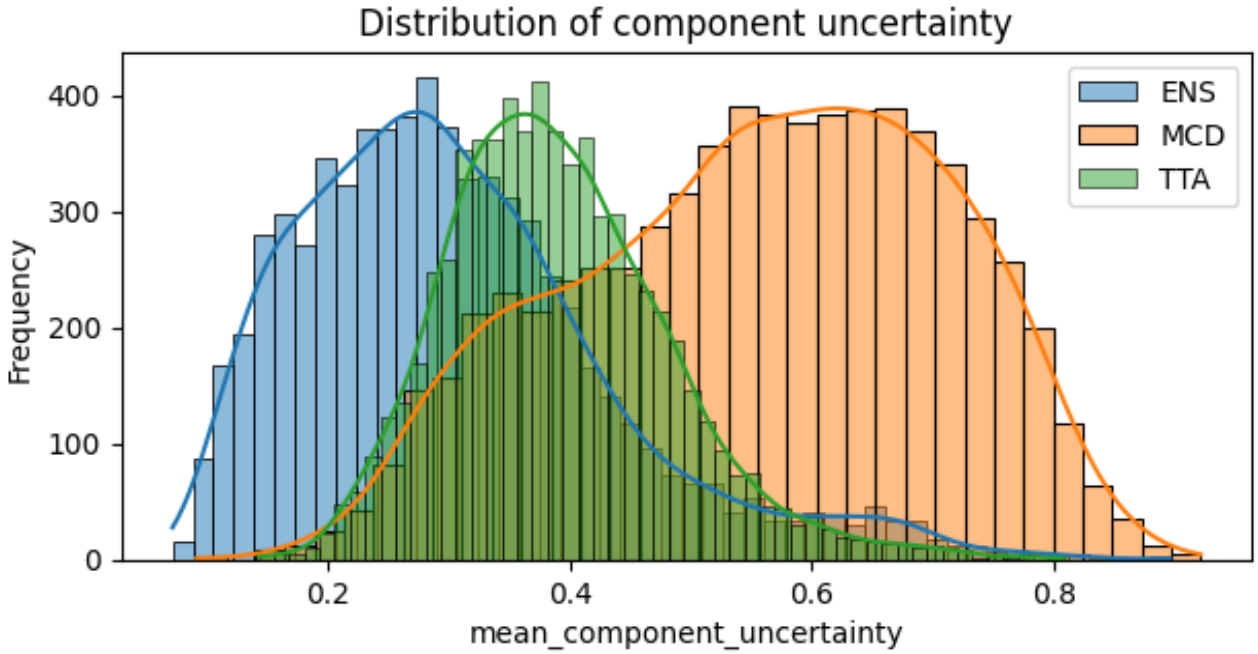


Figure 5.4 Component-level uncertainty distribution across data samples

Figure 5.4 shows the distributions of the component uncertainties. The graph shows that while TTA and ensemble methods show similar distributions, MC-dropout is slightly skewed to the right, showing higher uncertainty scores.

5.3.3 Image level Uncertainty

As explained before, the image level uncertainty is calculated using one of the metrics: Max, Mean, Weighted Mean, Thresholding, Weighted Thresholding. We needed to check the percentage of images under each threshold to use a valid threshold. Figure 5.5 shows the percentage of images under each threshold value on the x-axis. The graph was plotted using the mean uncertainty. Later, we show that the mean and weighted mean uncertainties have almost the same distribution. We chose a threshold to drop the top 20 percent of most uncertain data points. For MC-dropout, the threshold is 0.655, while for TTA, it is 0.431 and 0.359 for the ensemble.

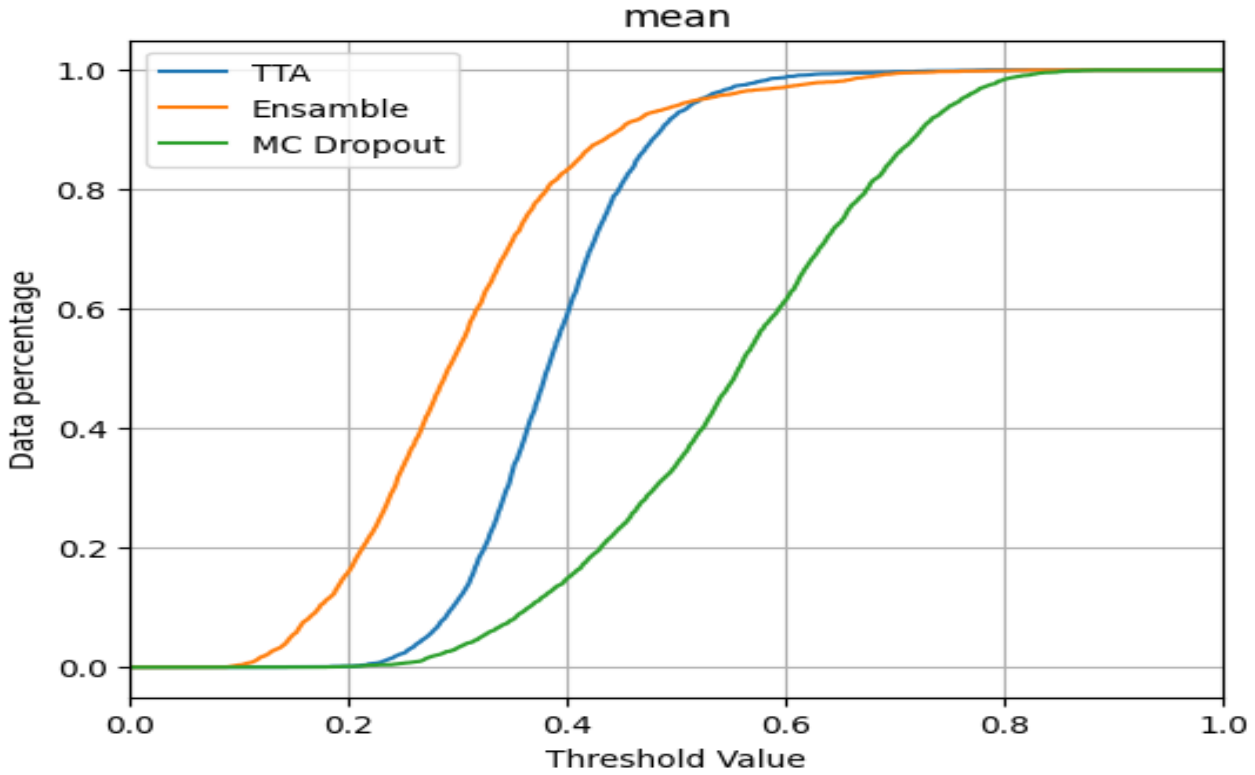


Figure 5.5 Percentage of images for uncertainty thresholds

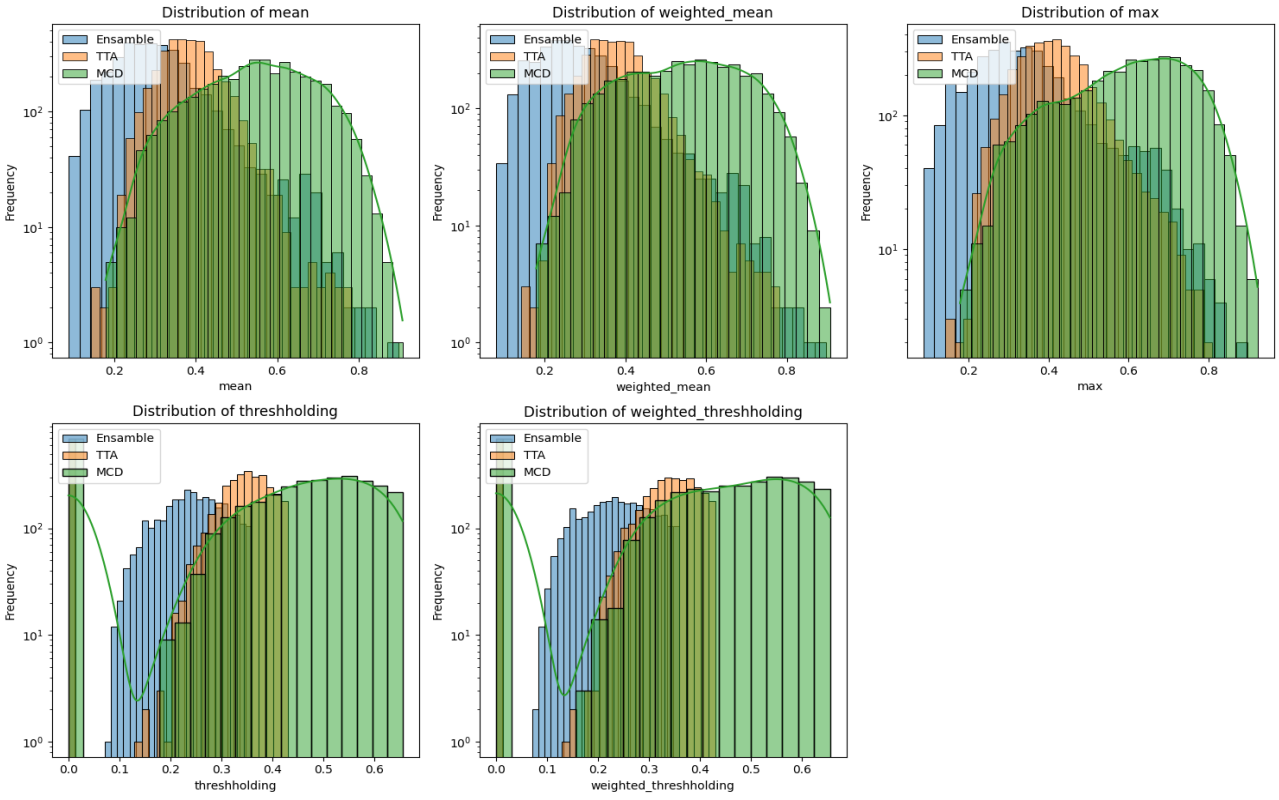


Figure 5.6 Image level uncertainty distribution across data samples for each metric

It can be observed from the graph that the mean and weighted mean, as well as thresholding and weighted thresholding, behave similarly, while the max pulls the uncertainty to higher values. Furthermore, the thresholding methods cause an increase in 0 uncertainties due to the images dropped for surpassing the uncertainty threshold. Hence, all thresholding methods have a cutoff.

5.3.4 Distribution overlap

To further decide how similar the distributions of each experiment are, we conducted an experiment showing the selection overlap. The overlap is calculated as the percentage of common samples selected by two UQ methods when selecting the top k highest uncertainty images using each method. In the following plots, The X-axis denotes the number of samples selected from the dataset; the Y-axis denotes the percentage of overlap (0 meaning no overlap, 1.0 meaning identical). The graph includes three plots for the identical, opposite, and random overlap, showing baselines for a best, worst, and random choice. First, we show that mean and weighted mean produce similar results; Figure 5.7 shows that all 3 UQ methods have over 80 percent overlap.

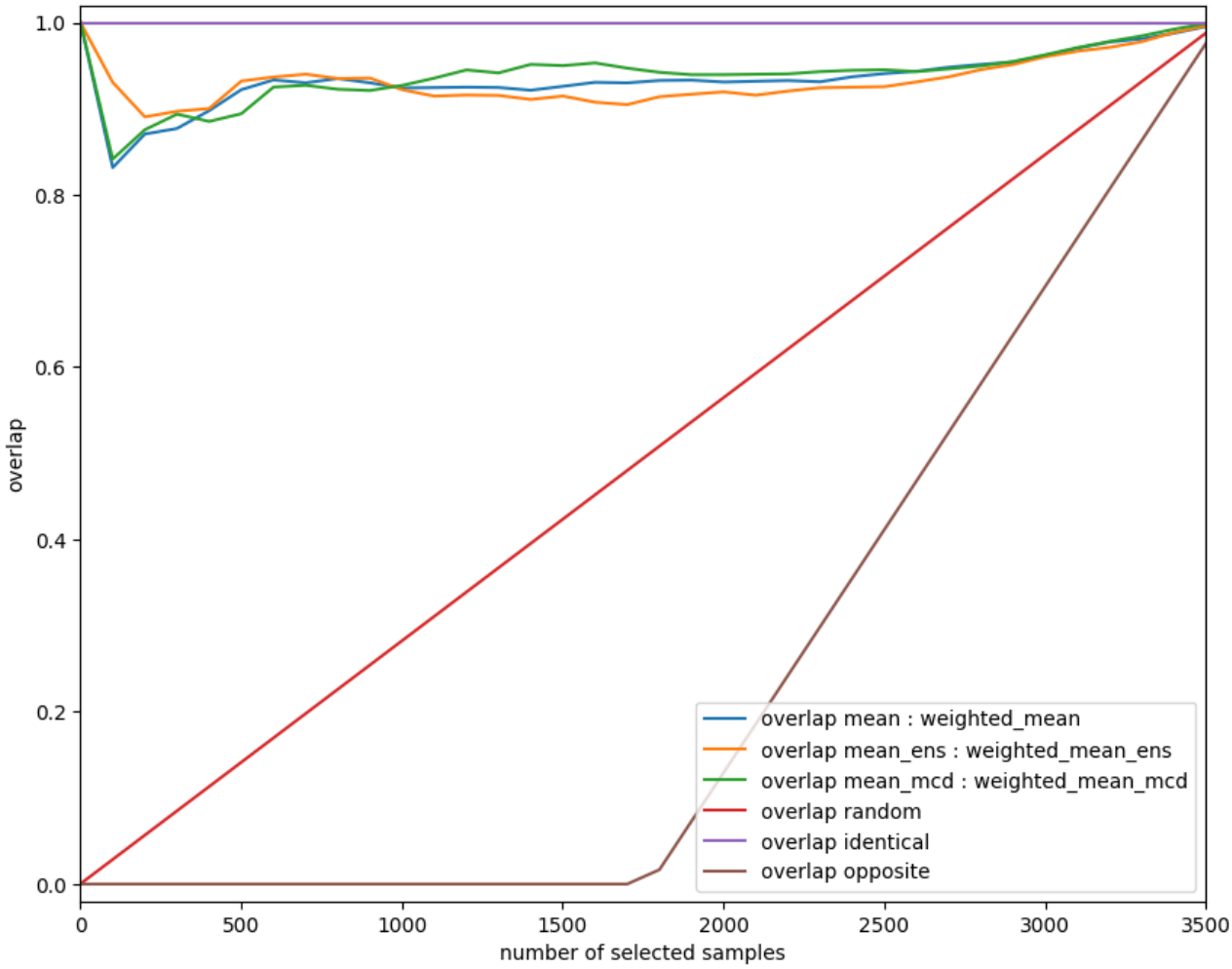


Figure 5.7 Overlap plot comparison of mean and weighted mean

We observe similar behavior between thresholding and weighted thresholding in Figure 5.8 where the overlap is over 90 percent.

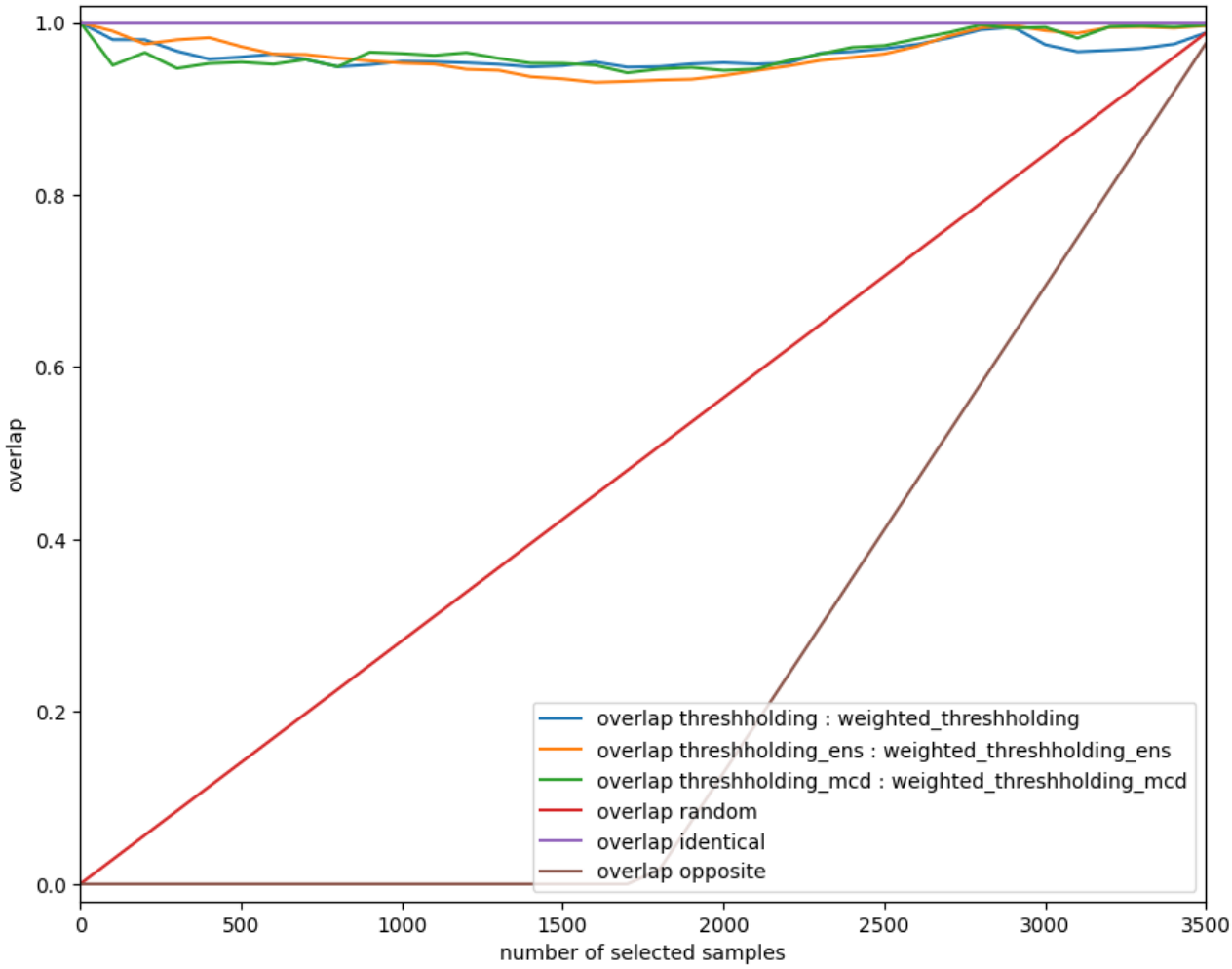


Figure 5.8 Overlap plot comparison of thresholding and weighted thresholding

For this reasoning, the comparison is made between weighted mean, max, and thresholding. Figure 5.11 shows the comparison of thresholding, while 5.9 shows the comparison using weighted mean, and finally 5.10 shows the results of using max.

The graphs show that the thresholding criterion, though it outperforms the random selection criteria, does not add a significant improvement. The max and mean criteria seem to be producing somewhat similar results. However, it is more plausible to utilize an aggregated mean, as for some images, a small object with high uncertainty might skew the entire image. In contrast, the mean would be more robust to such changes.

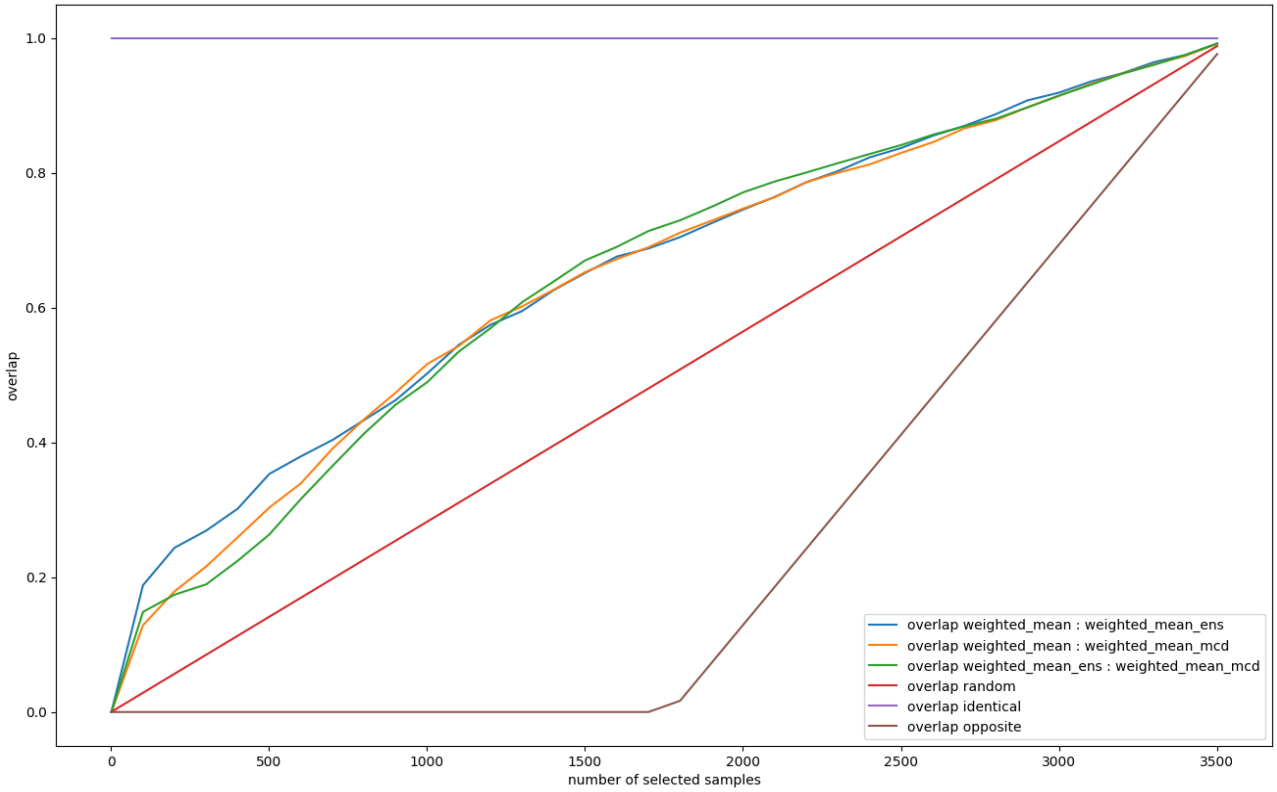


Figure 5.9 Overlap plot of weighted means

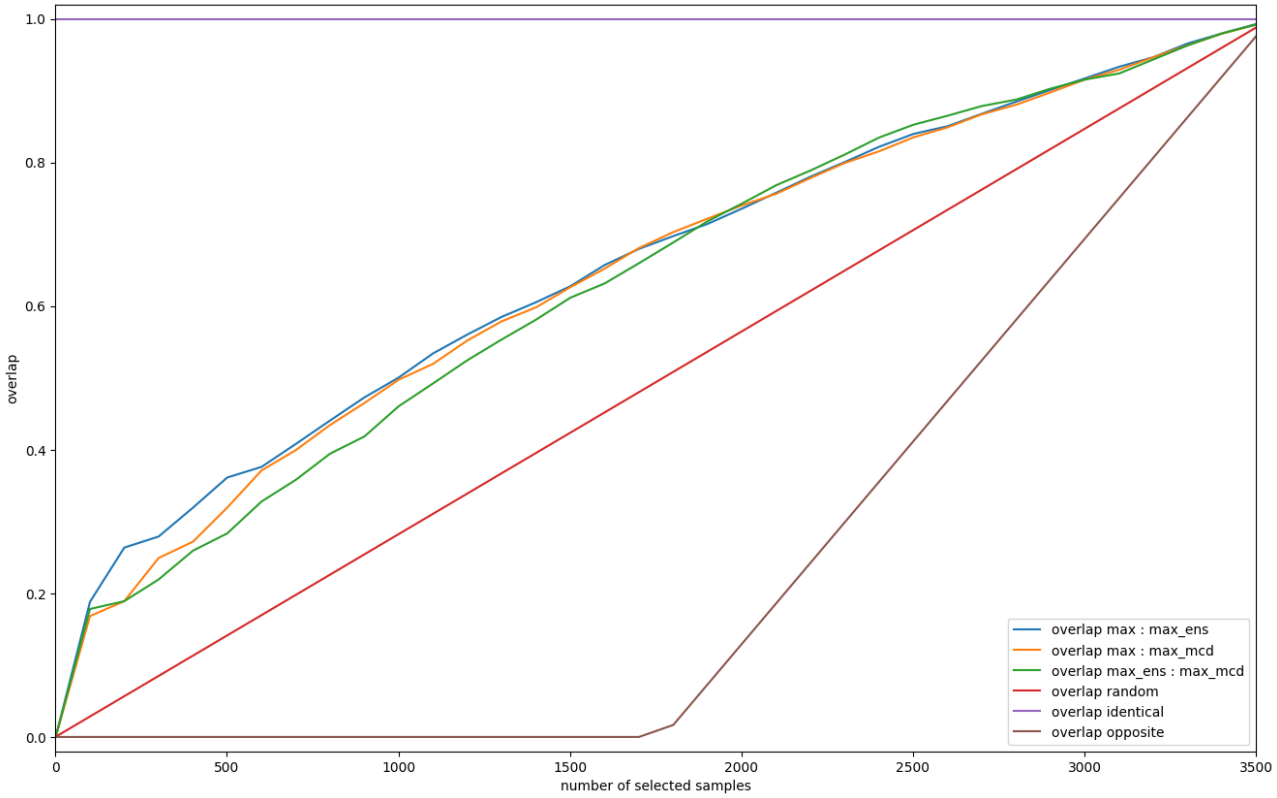


Figure 5.10 Overlap plot of max

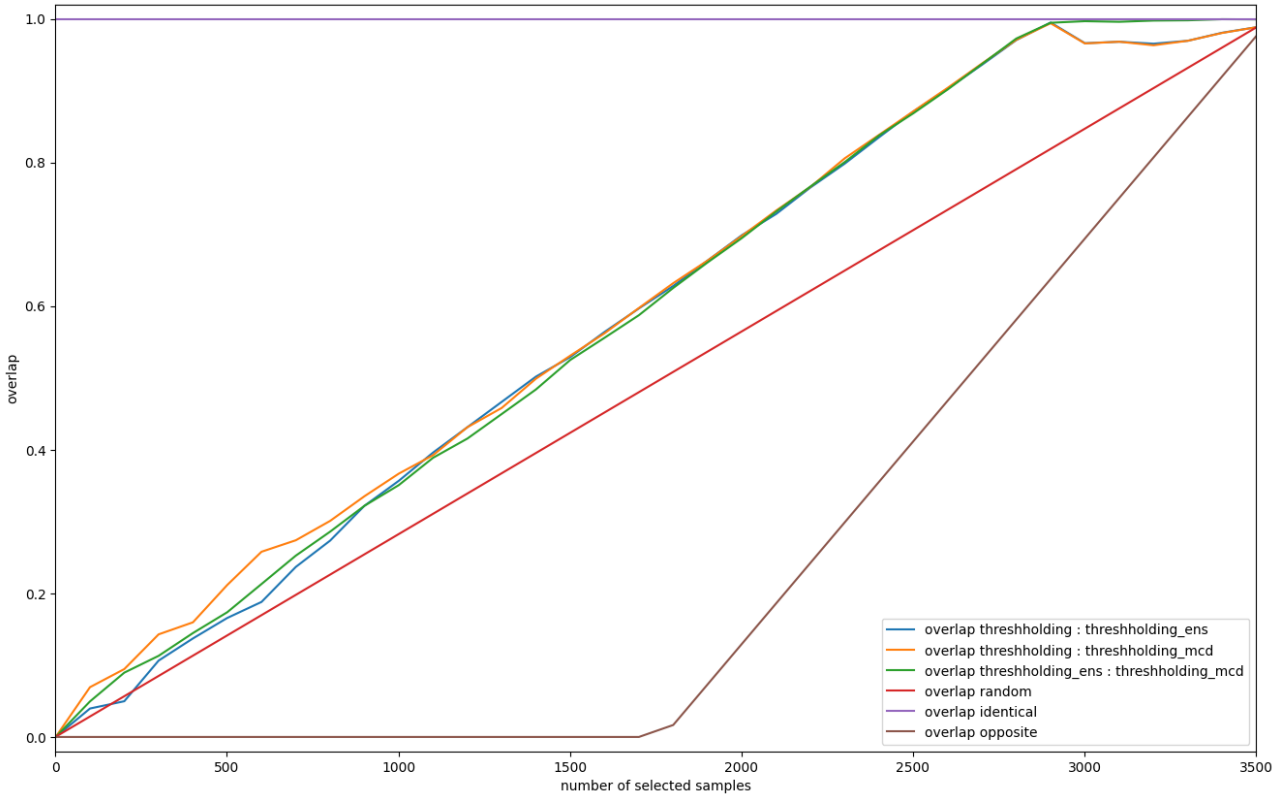


Figure 5.11 Overlap plot of thresholding

5.4 Active learning results

To compare the results of the AL iterations, the mAP of models after each iteration are compared. A baseline using a random selection of data points is used. Furthermore, due to time constraints, ensemble methods were not part of the AL experiments as they require training of each ensemble model for each iteration, amounting to multiple trained models. The comparison in this section lies between A random selection, MC-dropout, and the proposed TTA.

5.4.1 mAP

First, a visualization of each iteration is shown. Figure 5.12 shows the training iterations using each TTA. The green vertical lines denote the start of a new AL iteration, and the dashed horizontal lines denote the best mAP of each iteration, each one higher than the one before, denoting the improvement within each iteration, converging by the last iterations to minor improvements.

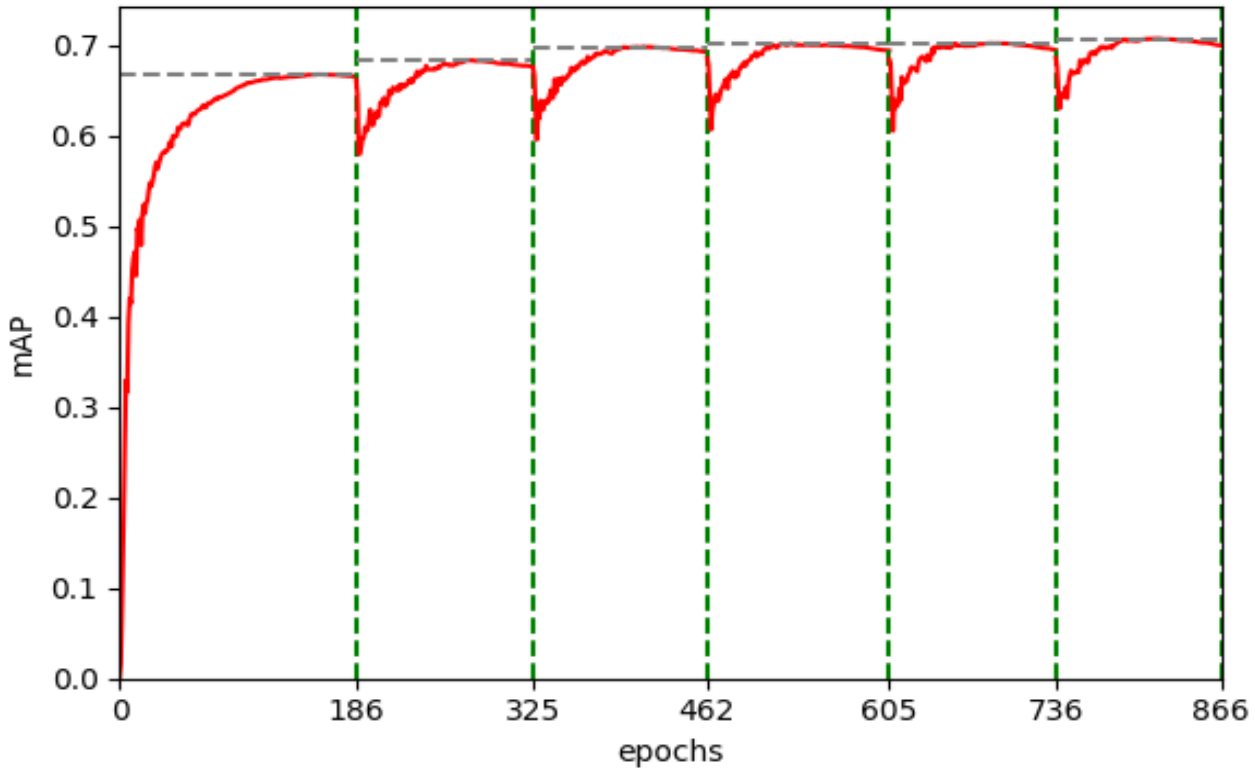


Figure 5.12 Active learning using TTA

The same plots for MC-dropout and random selection iterations are not shown since they're similar to the TTA plot above. However, a comparison of the mAP results of the models trained after each iteration can be seen in Figure 5.13. The X-axis denotes the AL iteration; the Y-axis denotes the mAP.

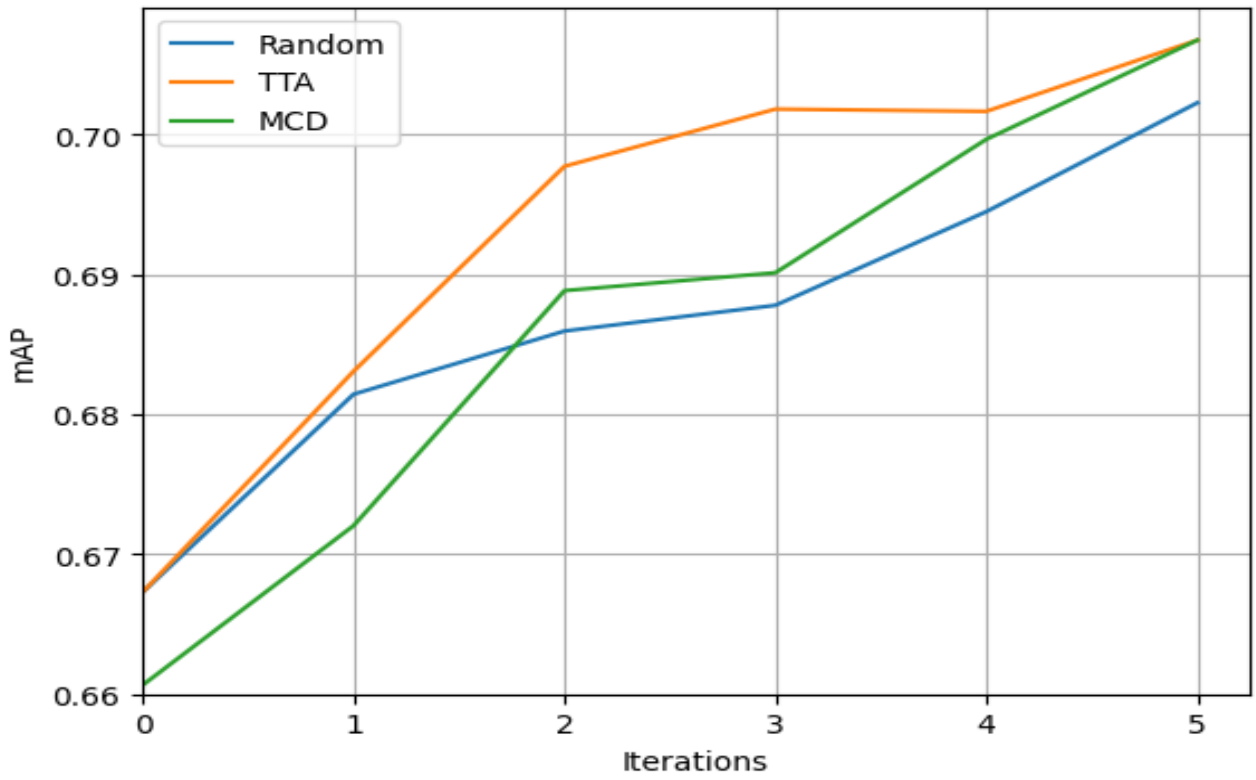


Figure 5.13 Comparison of the test-set mAP of each selection method

5.4.2 Uncertainty distribution

For each iteration, the most uncertain 5% of images were selected in addition to the training set. The following graphs show the uncertainty distribution of each image for each method used. Figure 5.14 shows the uncertainty distribution per iteration for TTA, while Figure 5.15 shows the distribution per iteration for MCD.

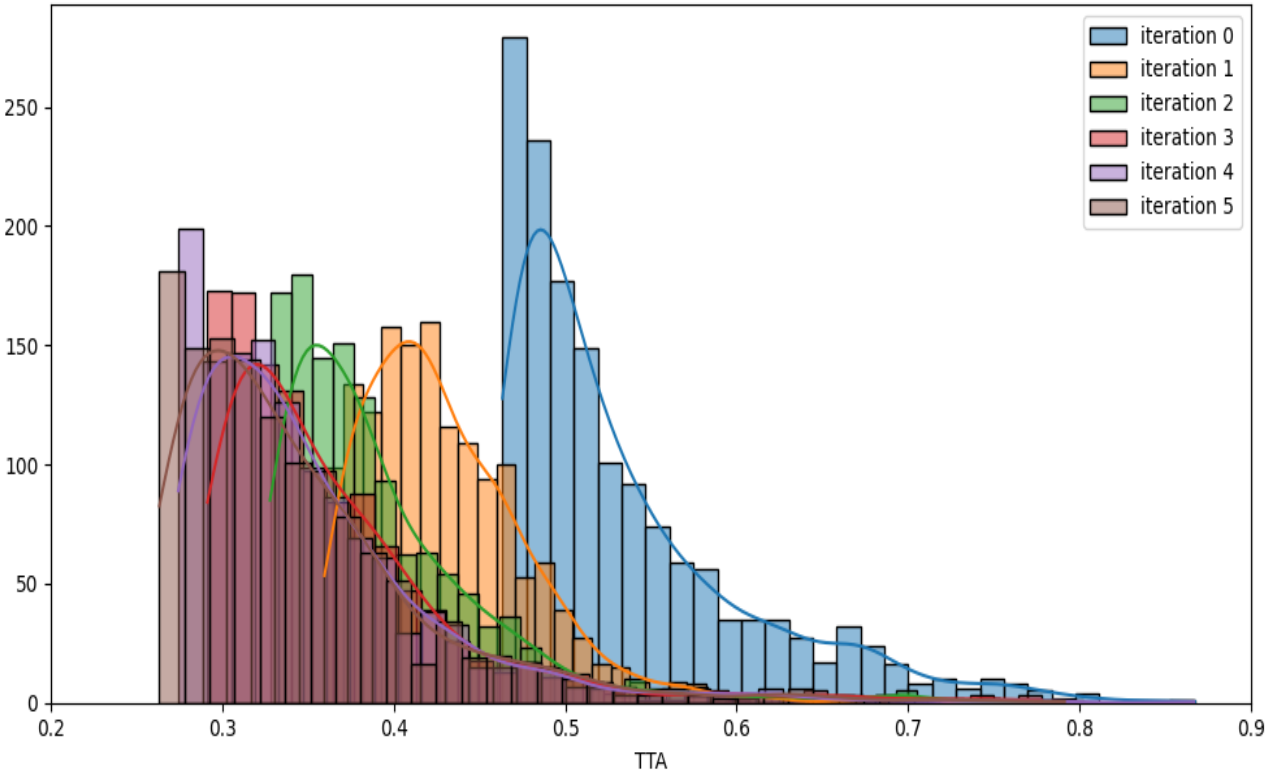


Figure 5.14 Uncertainty of most selected images for each iteration using TTA

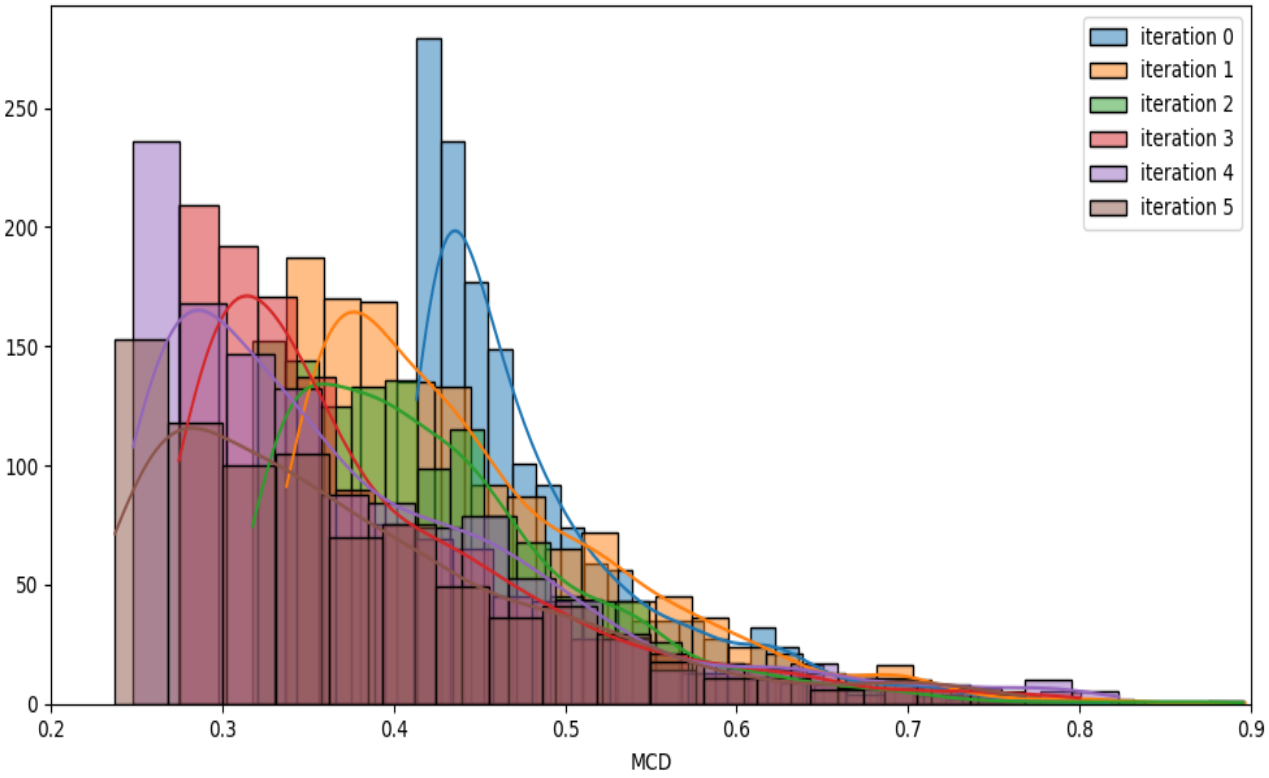


Figure 5.15 Uncertainty of most selected images for each iteration using MCD

It’s clear from both graphs that after each iteration, the total uncertainty is shifted to lower values, which shows success in AL iterations.

6 Discussion and conclusion

UQ is crucial in developing different systems. Specifically systems such as autonomous driving where complex and dynamic environments are met. Estimating uncertainties ensures reliable decision-making. This thesis proposes an efficient solution regarding memory and computation requirements, addressing some shortcomings of previous approaches. This thesis shows TTAs can be utilized as a Bayesian inference technique. The work in this thesis tackles the usage of augmentations, such as noise, blurring, or pixel value manipulations, on the input image as a batch prediction and using the output to estimate the uncertainty of the model. The proposed approach was tested on Yolov5 trained on the Mapillary dataset. Furthermore, the efficacy of TTAs was tested in AL experiments.

6.1 Limitations and Future work

Due to time limitations, comparing the ensemble methods in AL experiments could not be conducted. However, it would be insightful to observe the ensembles' behavior compared to the other methods. Furthermore, an analysis of the effect of changing the number of augmentations used and the augmentations mixed would be insightful to learn the optimal number of augmentations needed. In addition, a more in-depth comparison of the performance of the methods in terms of memory and time taken can provide clearer insights into how much better the performance boost of utilizing TTAs would be.

A further improvement area could be the optimization of component-level uncertainty calculations. In the current implementation, using anchors and matching them with all other boxes takes the longest time of the entire process. The more components are present in one image, the longer the estimation takes. Optimizing the calculations needed to produce an uncertainty score would be greatly beneficial.

On another note, all produced figures and graphs aimed to compare the total uncertainty of the model using different techniques. Even though aleatoric and epistemic uncertainties were also calculated, they were not compared across different methods. An analysis of different uncertainties could also be conducted.

It's also worth noting that the produced results could be implemented using different OD models and on different datasets to prove that the method can be widely adopted in further research.

Another question is whether to retrain models from previous iterations or train a new model from scratch in each new AL iteration.

6.2 Conclusion

In conclusion, this thesis has delved into utilizing TTAs to approximate the Bayesian inference technique for OD models. With AVs as a case study and a well-known dataset, the proposed method was compared against other SoTA methods. The results show that TTA succeeds as an uncertainty estimate technique for OD that is comparable to other methods. Furthermore, TTAs offer less computational cost and improved memory performance than the counterpart methods discussed. TTA does not require maintaining multiple models. Furthermore, it requires fewer forward passes. In addition, the augmentations can all be batched and vectorized into one forward pass, allowing for faster inference while maintaining similar uncertainty estimates for both counterparts.

A Appendix

The appendix section will be used for additional plots and results of different sections.

A.1 Pixel level uncertainty

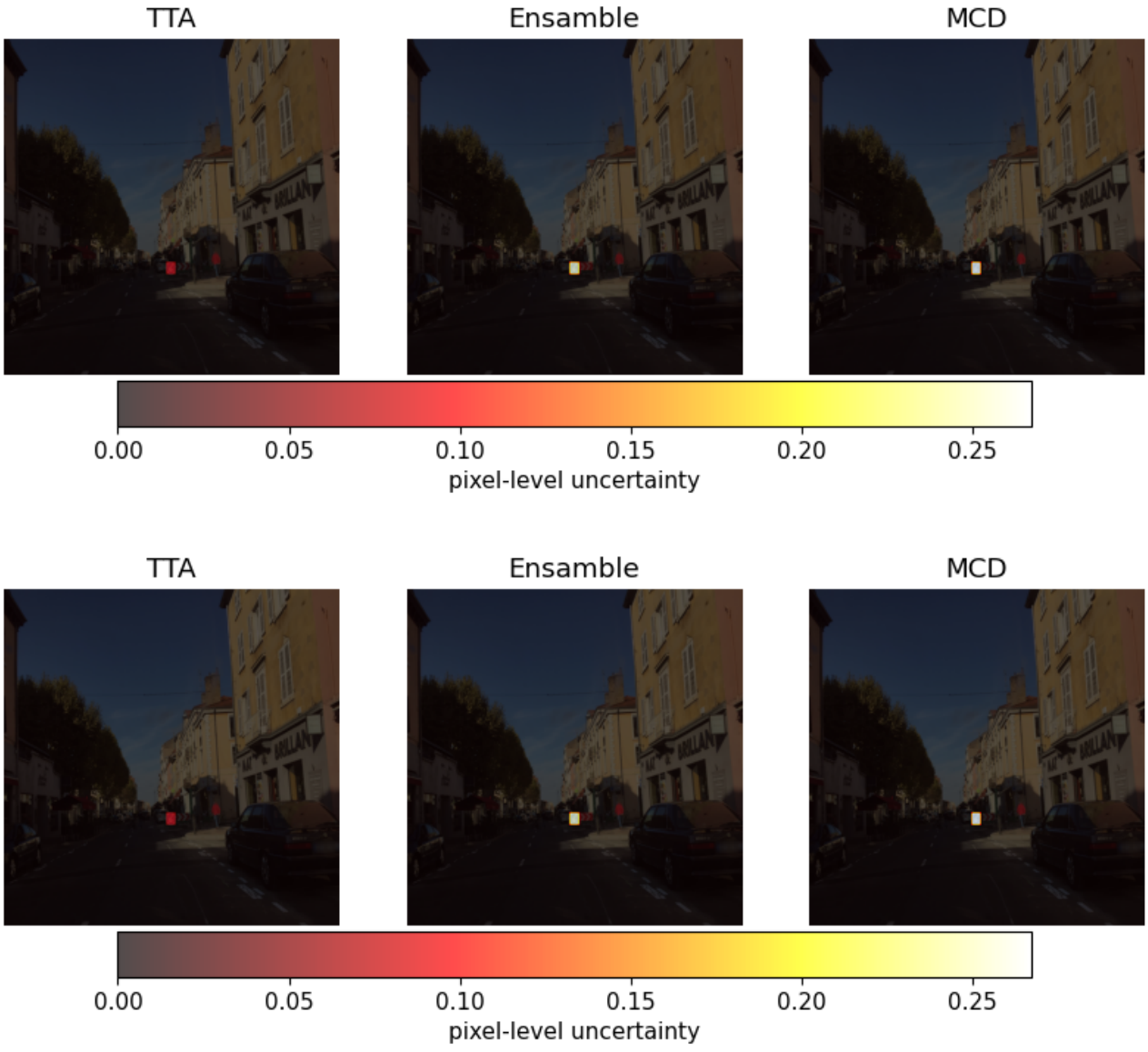


Figure A.1 Pixel level uncertainty estimation of different methods

A.2 Component level uncertainty

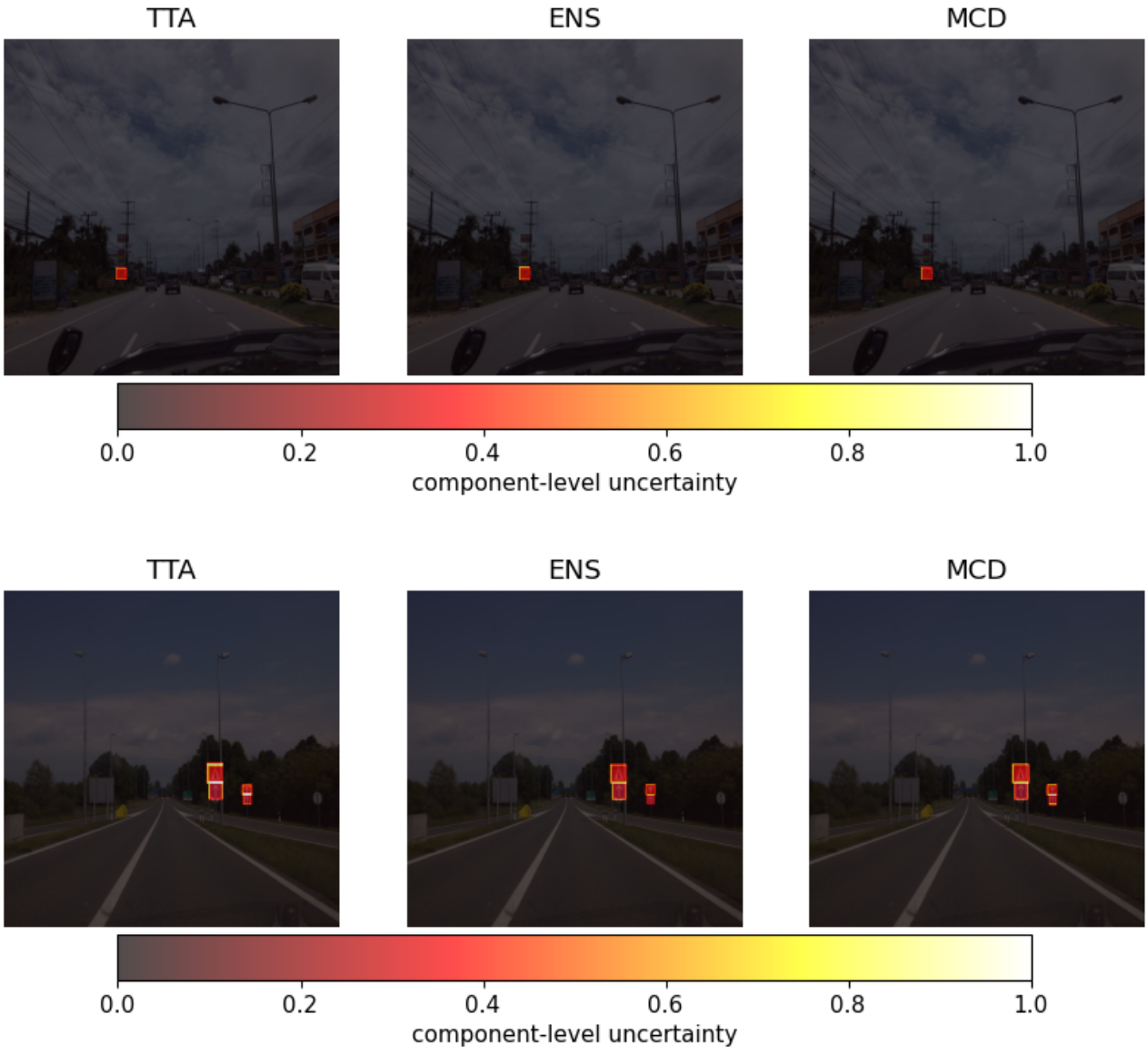


Figure A.2 Pixel level uncertainty estimation of different methods

A.3 Image level uncertainty

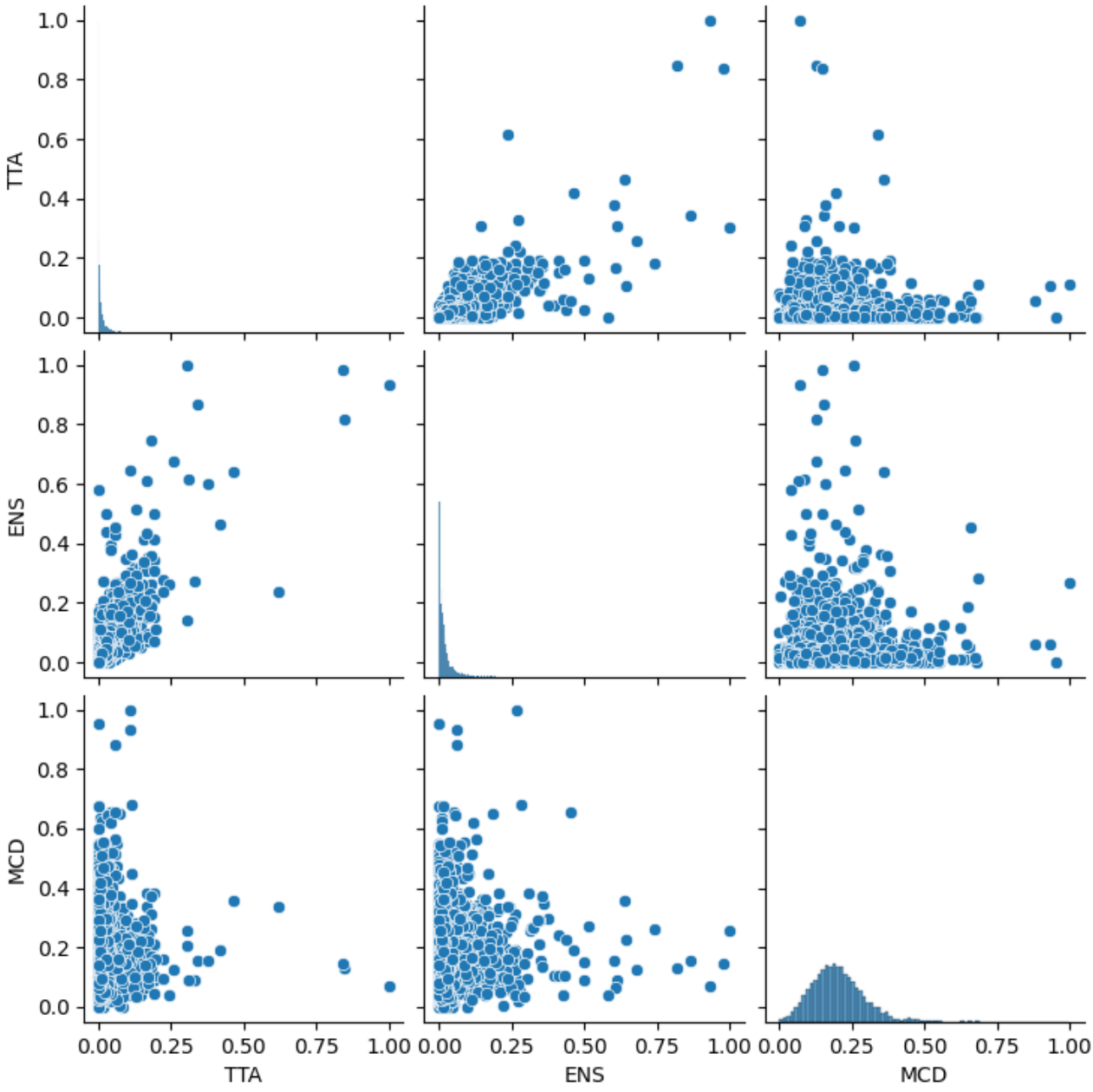


Figure A.3 Pair-plot of the image-level uncertainty estimates using all three methods calculated with mean pixel-wise uncertainty

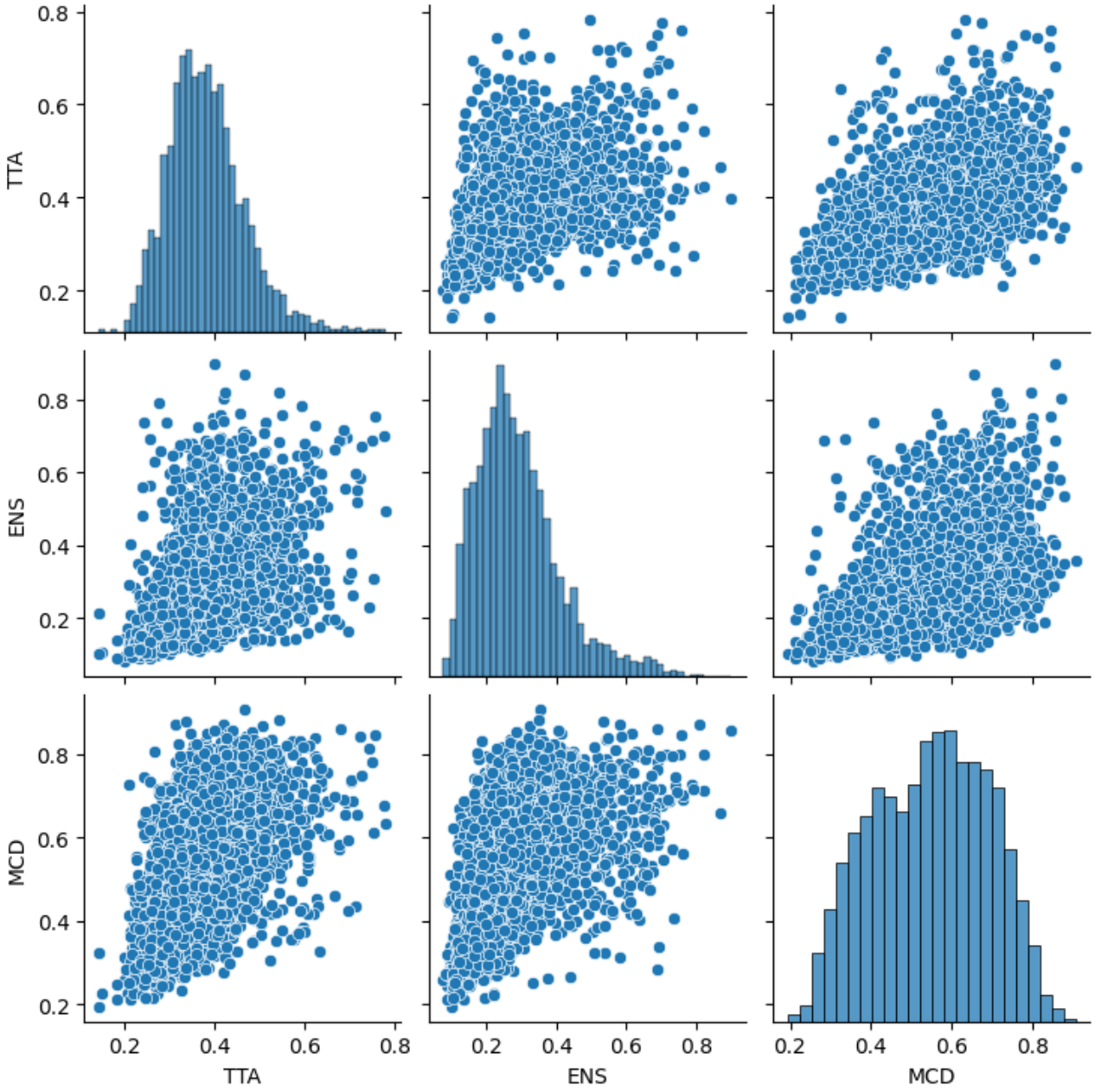


Figure A.4 Pair-plot of the image-level uncertainty estimates using all three methods calculated with weighted mean component-wise uncertainty

List of Figures

Acronyms

AI artificial intelligence. 1

AL active learning. 6, 9, 15, 24, 34, 35, 39

AV Autonomous vehicles. ix, 1, 11

BNN Bayesian neural network. 5, 7, 24

DL deep learning. 1, 8

DNN deep neural network. 3

GMM Gaussian mixture models. 8

IoU intersection over union. 18, 22

ML machine learning. 1, 4–6, 24

MVE mean-variance estimation. 8

NN neural network. 1, 7–9

OD object detection. 1, 3, 7, 9

SOTA state of the art. 2

TTA test-time augmentation. 1, 2, 8, 9, 20–22, 24, 39

TTD test-time dropout. 8

UQ uncertainty quantification. ix, 1, 2, 5–7, 9, 24, 31, 39

YOLO you only look once. 3

Bibliography

[1] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[2] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *International conference on machine learning*, pp. 843–852, PMLR, 2015.

[3] C. Krettek, "Chatgpt," *Die Unfallchirurgie*, vol. 126, no. 3, pp. 252–254, 2023.

[4] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58443–58469, 2020.

[5] M. Malik, M. K. Malik, K. Mehmood, and I. Makhdoom, "Automatic speech recognition: a survey," *Multimedia Tools and Applications*, vol. 80, pp. 9411–9457, 2021.

[6] J. G. Richens, C. M. Lee, and S. Johri, "Improving the accuracy of medical diagnosis with causal machine learning," *Nature communications*, vol. 11, no. 1, p. 3923, 2020.

[7] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International conference on machine learning*, pp. 1321–1330, PMLR, 2017.

[8] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information fusion*, vol. 76, pp. 243–297, 2021.

[9] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436, 2015.

[10] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *arXiv preprint arXiv:1610.02136*, 2016.

[11] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[12] C. Soize, *Uncertainty quantification*. Springer, 2017.

[13] T. J. Sullivan, *Introduction to uncertainty quantification*, vol. 63. Springer, 2015.

[14] M. Kimura, "Understanding test-time augmentation," in *International Conference on Neural Information Processing*, pp. 558–569, Springer, 2021.

[15] D. Shanmugam, D. Blalock, G. Balakrishnan, and J. Guttag, "Better aggregation in test-time augmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1214–1223, 2021.

[16] P. Bharati and A. Pramanik, "Deep learning techniques—r-cnn to mask r-cnn: a survey," *Computational Intelligence in Pattern Recognition: Proceedings of CIPR 2019*, pp. 657–668, 2020.

[17] B. Cheng, Y. Wei, H. Shi, R. Feris, J. Xiong, and T. Huang, "Revisiting rcnn: On awakening the classification power of faster rcnn," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 453–468, 2018.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[19] H.-C. Chen, A. M. Widodo, A. Wisnujati, M. Rahaman, J. C.-W. Lin, L. Chen, and C.-E. Weng, "Alexnet convolutional neural network for disease detection and classification of tomato leaf," *Electronics*, vol. 11, no. 6, p. 951, 2022.

[20] H. Jiang and E. Learned-Miller, "Face detection with the faster r-cnn," in *2017 12th IEEE international conference on automatic face & gesture recognition (FG 2017)*, pp. 650–657, IEEE, 2017.

[21] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pp. 21–37, Springer, 2016.

[23] M. Gong, D. Wang, X. Zhao, H. Guo, D. Luo, and M. Song, "A review of non-maximum suppression algorithms for deep learning target detection," in *Seventh Symposium on Novel Photoelectronic Detection Technology and Applications*, vol. 11763, pp. 821–828, SPIE, 2021.

[24] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *18th international conference on pattern recognition (ICPR'06)*, vol. 3, pp. 850–855, IEEE, 2006.

[25] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[26] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, and D. Menotti, "A robust real-time automatic license plate recognition based on the yolo detector," in *2018 international joint conference on neural networks (ijcnn)*, pp. 1–10, IEEE, 2018.

[27] Y. Tian, G. Yang, Z. Wang, H. Wang, E. Li, and Z. Liang, "Apple detection during different growth stages in orchards using the improved yolo-v3 model," *Computers and electronics in agriculture*, vol. 157, pp. 417–426, 2019.

[28] G. Jocher, "YOLOv5 by Ultralytics," May 2020.

[29] F. Dang, D. Chen, Y. Lu, and Z. Li, "Yoloweeds: a novel benchmark of yolo object detectors for multi-class weed detection in cotton production systems," *Computers and Electronics in Agriculture*, vol. 205, p. 107655, 2023.

[30] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of yolo algorithm developments," *Procedia Computer Science*, vol. 199, pp. 1066–1073, 2022.

[31] A. Der Kiureghian and O. Ditlevsen, "Aleatory or epistemic? does it matter?," *Structural safety*, vol. 31, no. 2, pp. 105–112, 2009.

[32] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," *Advances in neural information processing systems*, vol. 30, 2017.

[33] G. Freni and G. Mannina, "Bayesian approach for uncertainty quantification in water quality modelling: The influence of prior distribution," *Journal of Hydrology*, vol. 392, no. 1-2, pp. 31–39, 2010.

[34] Y. Gal, P. Koumoutsakos, F. Lanusse, G. Louppe, and C. Papadimitriou, “Bayesian uncertainty quantification for machine-learned models in physics,” *Nature Reviews Physics*, vol. 4, no. 9, pp. 573–577, 2022.

[35] T. A. Mara, F. Delay, F. Lehmann, and A. Younes, “A comparison of two bayesian approaches for uncertainty quantification,” *Environmental Modelling & Software*, vol. 82, pp. 21–30, 2016.

[36] R. Rahaman *et al.*, “Uncertainty quantification and deep ensembles,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 20063–20075, 2021.

[37] L. Hoffmann, I. Fortmeier, and C. Elster, “Uncertainty quantification by ensemble learning for computational optical form measurements,” *Machine Learning: Science and Technology*, vol. 2, no. 3, p. 035030, 2021.

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

[39] L. L. Folgoc, V. Baltatzis, S. Desai, A. Devaraj, S. Ellis, O. E. M. Manzanera, A. Nair, H. Qiu, J. Schnabel, and B. Glocker, “Is mc dropout bayesian?,” *arXiv preprint arXiv:2110.04286*, 2021.

[40] L. P. Swiler, T. L. Paez, and R. L. Mayes, “Epistemic uncertainty quantification tutorial,” in *Proceedings of the 27th International Modal Analysis Conference*, 2009.

[41] J. Jakeman, M. Eldred, and D. Xiu, “Numerical approach for quantification of epistemic uncertainty,” *Journal of Computational Physics*, vol. 229, no. 12, pp. 4648–4663, 2010.

[42] S. Lahlou, M. Jain, H. Nekoei, V. I. Butoi, P. Bertin, J. Rector-Brooks, M. Korablyov, and Y. Bengio, “Deup: Direct epistemic uncertainty prediction,” *arXiv preprint arXiv:2102.08501*, 2021.

[43] M. Monteiro, L. Le Folgoc, D. Coelho de Castro, N. Pawlowski, B. Marques, K. Kamnitsas, M. van der Wilk, and B. Glocker, “Stochastic segmentation networks: Modelling spatially correlated aleatoric uncertainty,” *Advances in neural information processing systems*, vol. 33, pp. 12756–12767, 2020.

[44] B. Adlam, J. Snoek, and S. L. Smith, “Cold posteriors and aleatoric uncertainty,” *arXiv preprint arXiv:2008.00029*, 2020.

[45] A. Mavor-Parker, K. Young, C. Barry, and L. Griffin, “How to stay curious while avoiding noisy tvs using aleatoric uncertainty estimation,” in *International Conference on Machine Learning*, pp. 15220–15240, PMLR, 2022.

[46] C. Brame, “Active learning,” *Vanderbilt University Center for Teaching*, 2016.

[47] B. Settles, “Active learning literature survey,” 2009.

[48] R. Burbidge, J. J. Rowland, and R. D. King, “Active learning for regression based on query by committee,” in *Intelligent Data Engineering and Automated Learning-IDEAL 2007: 8th International Conference, Birmingham, UK, December 16-19, 2007. Proceedings 8*, pp. 209–218, Springer, 2007.

[49] H. Hino and S. Eguchi, “Active learning by query by committee with robust divergences,” *Information Geometry*, vol. 6, pp. 81–106, 2023.

[50] V.-L. Nguyen, M. H. Shaker, and E. Hüllermeier, “How to measure uncertainty in uncertainty sampling for active learning,” *Machine Learning*, vol. 111, pp. 89–122, 2022.

[51] J. Zhu, H. Wang, B. K. Tsou, and M. Ma, “Active learning with sampling by uncertainty and density for data annotations,” *IEEE Transactions on audio, speech, and language processing*, vol. 18, pp. 1323–1331, 2009.

[52] J. M. Bernardo and A. F. Smith, *Bayesian theory*, vol. 405. John Wiley & Sons, 2009.

[53] I. Kononenko, "Bayesian neural networks," *Biological Cybernetics*, vol. 61, no. 5, pp. 361–370, 1989.

[54] V. Mullachery, A. Khera, and A. Husain, "Bayesian neural networks," *arXiv preprint arXiv:1801.07710*, 2018.

[55] B. Kerleguer, C. Cannamela, and J. Garnier, "A bayesian neural network approach to multi-fidelity surrogate modeling," *International Journal for Uncertainty Quantification*, vol. 14, no. 1, 2024.

[56] D. R. Farine and A. Strandburg-Peshkin, "Estimating uncertainty and reliability of social network data using bayesian inference," *Royal Society open science*, vol. 2, no. 9, p. 150367, 2015.

[57] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[58] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," *Advances in neural information processing systems*, vol. 28, 2015.

[59] S.-i. Maeda, "A bayesian encourages dropout," *arXiv preprint arXiv:1412.7003*, 2014.

[60] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems*, vol. 30, 2017.

[61] J. Liu, J. Paisley, M.-A. Kioumourtzoglou, and B. Coull, "Accurate uncertainty estimation and decomposition in ensemble learning," *Advances in neural information processing systems*, vol. 32, 2019.

[62] F. Wenzel, J. Snoek, D. Tran, and R. Jenatton, "Hyperparameter ensembles for robustness and uncertainty quantification," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6514–6527, 2020.

[63] A. Khosravi and S. Nahavandi, "An optimized mean variance estimation method for uncertainty quantification of wind power forecasts," *International Journal of Electrical Power & Energy Systems*, vol. 61, pp. 446–454, 2014.

[64] X. Huang, T. Zhao, and S. Kudratova, "Uncertain mean-variance and mean-semivariance models for optimal project selection and scheduling," *Knowledge-Based Systems*, vol. 93, pp. 1–11, 2016.

[65] D. A. Reynolds *et al.*, "Gaussian mixture models.," *Encyclopedia of biometrics*, vol. 741, no. 659-663, 2009.

[66] C. Viroli and G. J. McLachlan, "Deep gaussian mixture models," *Statistics and Computing*, vol. 29, pp. 43–51, 2019.

[67] A. Amini, W. Schwarting, A. Soleimany, and D. Rus, "Deep evidential regression," *Advances in Neural Information Processing Systems*, vol. 33, pp. 14927–14937, 2020.

[68] N. Meinert, J. Gawlikowski, and A. Lavin, "The unreasonable effectiveness of deep evidential regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 9134–9142, 2023.

[69] N. Meinert and A. Lavin, "Multivariate deep evidential regression," *arXiv preprint arXiv:2104.06135*, 2021.

[70] M. S. Ayhan and P. Berens, "Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks," in *Medical Imaging with Deep Learning*, 2022.

[71] G. Wang, W. Li, M. Aertsen, J. Deprest, S. Ourselin, and T. Vercauteren, “Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks,” *Neurocomputing*, vol. 338, pp. 34–45, 2019.

[72] K. Patel, W. Beluch, D. Zhang, M. Pfeiffer, and B. Yang, “On-manifold adversarial data augmentation improves uncertainty calibration,” in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 8029–8036, IEEE, 2021.

[73] F. Kraus and K. Dietmayer, “Uncertainty estimation in one-stage object detection,” in *2019 ieee intelligent transportation systems conference (itsc)*, pp. 53–60, IEEE, 2019.

[74] S. Gasperini, J. Haug, M.-A. N. Mahani, A. Marcos-Ramiro, N. Navab, B. Busam, and F. Tombari, “Certainnet: Sampling-free uncertainty estimation for object detection,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 698–705, 2021.

[75] Z. Wang, D. Feng, Y. Zhou, L. Rosenbaum, F. Timm, K. Dietmayer, M. Tomizuka, and W. Zhan, “Inferring spatial uncertainty in object detection,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5792–5799, IEEE, 2020.

[76] M. Schubert, K. Kahl, and M. Rottmann, “Metadetect: Uncertainty quantification and prediction quality estimates for object detection,” in *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10, IEEE, 2021.

[77] C. Ertler, J. Mislej, T. Ollmann, L. Porzi, G. Neuhold, and Y. Kuang, “The mapillary traffic sign dataset for detection and classification on a global scale,” in *European Conference on Computer Vision*, pp. 68–84, Springer, 2020.

[78] D. M. Hawkins, “The problem of overfitting,” *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[79] V. Feldman, R. Frostig, and M. Hardt, “The advantages of multiple classes for reducing overfitting from test set reuse,” in *International Conference on Machine Learning*, pp. 1892–1900, PMLR, 2019.

[80] R. Padilla, S. L. Netto, and E. A. Da Silva, “A survey on performance metrics for object-detection algorithms,” in *2020 international conference on systems, signals and image processing (IWSSIP)*, pp. 237–242, IEEE, 2020.

[81] P. Henderson and V. Ferrari, “End-to-end training of object class detectors for mean average precision,” in *Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part V 13*, pp. 198–213, Springer, 2017.