Master's thesis
Master's Programme in Data Science

Localization Aware Active Learning for IoU Predicting Object Detectors

Mikko Saukkoriipi

March 22, 2022

Supervisor(s): Associate Professor Laura Ruotsalainen

Examiner(s): Associate Professor Laura Ruotsalainen
Professor Jukka K. Nurminen

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Mikko Saukkoriipi

Työn nimi — Arbetets titel — Title

Localization Aware Active Learning for IoU Predicting Object Detectors

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	March 22, 2022	65

Tiivistelmä — Referat — Abstract

Two factors define the success of a deep neural network (DNN) based application; the training data and the model. Nowadays, many state-of-the-art DNN models are available free of charge, and training and deploying these models is easier than ever before. As a result, anyone can set up a state-of-the-art DNN algorithm within days or even hours.

In the past, most of the focus has been given to the model when researchers were building faster and more accurate deep learning architectures. These research groups commonly use large and high-quality datasets in their work, which is not the case when one wants to train a new model for a specific use case.

Training a DNN algorithm for a specific task requires collecting a vast amount of unlabelled data and then labeling the training data. To train a high-performance model, the labeled training dataset must be large and diverse to cover all relevant scenarios of the intended use case.

This thesis will present an efficient and straightforward active learning method to sample the most informative images to train a powerful anchor-free Intersection over Union (IoU) predicting objector detector. Our method only uses classification confidences and IoU predictions to estimate the image informativeness. By collecting the most informative images, we can cover the whole diversity of the images with fewer human-annotated training images. This will save time and resources, as we avoid labeling images that would not be beneficial.

ACM Computing Classification System (CCS):
Computing methodologies → Artificial intelligence → Computer vision → Object detection
Computing methodologies → Machine learning → Learning settings → Active learning settings

Avainsanat — Nyckelord — Keywords

computer vision, active learning, object detection

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1 Introduction **1**

1.1 Motivation of the research . . . 3

1.2 Related work . . . 3

1.3 Structure of thesis . . . 4

2 Convolutional Neural networks in Object Detection **7**

2.1 Object Detector Architecture . . . 8

2.2 Data augmentation . . . 11

2.3 Real-time and slower object detectors . . . 13

3 Performance Metrics for Object-Detection Algorithms **17**

3.1 Definition of correct localization . . . 17

3.2 Precision and Recall . . . 19

3.3 Average Precision (AP) and Mean Average Precision (mAP) . . . 20

4 YOLOX **23**

4.1 YOLO detectors history . . . 24

4.2 YOLOX - latest innovations to YOLO architecture . . . 30

4.3 YOLOX Performance . . . 33

5 Active Learning **35**

5.1 Active Learning vs. Passive Learning . . . 36

5.2 Active Learning Process . . . 37

5.2.1 Defining the seed set . . . 39

5.2.2 Selecting the most informative data samples . . . 39

5.2.3 Terminating the active learning loop . . . 41

5.3 Active learning as annotation support . . . 42

6 Experiments **45**

6.1 Pascal VOC object detection dataset . . . 45

6.2 YOLOX-S object detector . . . 47

6.3 Model training 47
6.4 Defining detection uncertainty score 49
6.5 Used Active Learning Process 49
6.6 Training rounds progress 51
6.7 Results 53

7 Conclusions **59**

Bibliography **61**

1. Introduction

Two factors define the success of a deep learning-based application; the training data and the model. In the past, most of the focus has been given to the model when researchers were building faster and more accurate deep learning architectures. Commonly these research groups use large and high-quality datasets in their work. In object detection, these are Pascal Visual Object Classes [8] or COCO dataset [25].

Nowadays, it is easier than ever before to create and deploy a deep learning model for any task. Many of the latest state-of-the-art deep learning models are available free of charge and with easy-to-use instructions on the webpages like Pytorch Hub [2], Tensorflow Hub [3] and Model Zoo [1]. Because of this, it is possible to set up a state-of-the-art deep learning model in just a few days or even within hours.

As many ready-made models are available, the major work lies in creating the training data. Even there are available a wide variety of ready-made training datasets, these datasets only cover a small subset of all possible use cases. Therefore it is expected that there is no ready-made dataset available for one's use case, and one needs to collect and annotate his training dataset.

Annotating the training data is especially expensive for the object detection task [20]. In the object detection task, it is common that a single image includes multiple objects. In the annotation process, the human annotator needs to draw a tight box bounding box around each object in the image and then classify the content of each bounding box. This process is significantly more difficult and time-consuming than annotating training data for the image classification task, where the annotator can answer multiple-choice questions of which class the image belongs to [39]. At the same time, it is also more challenging to monitor the annotation quality.

In the context of object detection, active learning is not very well studied, and estimating image informativeness is a challenging task [20]. The challenge comes from the fact that the object detection model does two tasks: object localization and classification. Because of the two subtasks settings, we need to assess if the unlabelled image has new information to improve localization, classification, or both [6]. As the images are not annotated, we need to do this estimation without knowing the ground truth labels.

In my master's thesis, I will research active learning in the context of object detection with convolutional neural networks. We focus on the object detectors that output classification confidences, predict estimated bounding box Intersection over Union (IoU), and use anchor-free localization. These properties can be seen as a new paradigm in object detectors, as the first IoU predicting object detector was released in 2020 [21], and since that, they have been successfully implemented to other object detectors like You Only Look Once version X (YOLOX) [11].

The work aims to find an efficient and automated way to select the most informative images for the human annotator and that way speed up the development process and save resources. My master's thesis focuses on active learning with massive data volumes, which requires used techniques to be computationally efficient. We use the YOLOX object detector in our experiment [11].

Previous research on this topic is not scalable to massive data volumes [20], or the active learning method only uses classification confidences when estimating image informativity [6] [14]. In this work, we propose a maximum uncertainty active learning method for anchor-free object detection algorithms that natively predict the estimated IoU, which can be used to estimate the localization confidence of the detected objects. Our proposed method only uses classification confidences and predicted IoU to define the image informativity score, and it does not require any changes to the object detector architecture. As a result, the proposed method is computationally efficient, and it uses both classification and localization metrics when measuring image informativity.

As a summary, we want to answer the following questions with the thesis:

1. How can we minimize the number of required training images to train a modern object detection algorithm?
2. Can we leverage object detector predicted IoU to select the most informative unlabeled images?

To make the proposed method useful in real-life use cases, it needs to fulfill the following requirements:

1. The proposed method needs to be computationally efficient.
2. The proposed sampling method must keep the training dataset balanced (sample images from each object class).

1.1 Motivation of the research

To our knowledge, this is the first active learning research in object detection, where the model predicted IoU is used to estimate the localization confidence. Secondly, the previous methods have given only a little or no attention to keeping the training dataset balanced during the active learning process [14][6][20]. If the training dataset is not kept balanced, it may hurt the model's performance to detect an object from the rarest object classes [42].

The model predicted IoU has been proved to be an efficient way to estimate the localization confidence. This is proved by Kim et al., where IoU prediction was found to have on average less than 0.1 error, and test was done with multiple different backbones [21]. Therefore we can argue that IoU prediction can be used to estimate the localization confidence, which would benefit the active learning process to sample the most informative unlabelled images. However, this has topic has not been studied before in the context of active learning.

The challenge of the unbalanced dataset is common when working with usual object detection datasets like COCO and Pascal VOC, but they are very common in real life [42]. Also, it is expected that a vast number of unlabelled images do not have any objects in real life, and we want to filter out these images in the active learning process.

An unbalanced dataset changes the strategy of how most informative images should be sampled. If the training dataset is not kept balanced, it is likely that even if the model's overall performance improves, the performance to detect and classify the rarest classes may stay poor [42]. We wanted to keep the dataset as balanced as possible through the active learning process in our method. In practice, this means that we aim to select 100 most informative images for each 20 object class on each active learning round. Therefore our method is designed by default to keep training dataset classes balanced.

1.2 Related work

The research by Kao et al. 2018 investigates active learning with a two-stage object detector Faster R-CNN [20]. Their method achieved the best results by using localization stability and classification uncertainty to estimate image informativity. However, localization stability is measured by comparing object localization in the image to localizations with the same image added with Gaussian noise. This means that every unlabelled image must be processed twice, which causes a computational overhead. Besides, we expect that using Gaussian noise to estimate localization stability would

be less efficient when strong data augmentation is used in the model training process because strong data augmentation increases model robustness against adversarial examples and corrupted and out-of-distribution inputs [44][43].

In the research by Brust et al. 2018 they study active learning in object detection by calculating scores based on the classification confidence distribution [6]. The experiment uses a YOLOv3 object detector [34], which uses a logistic classifier to classify objects. This means that for each classified object, a logistic classifier will give confidence to each possible class. Then the image informativity score is calculated by taking the absolute difference between the highest and second-highest class scores for each detected object. The theory behind this technique is that if the difference between the two highest class scores is very low, then it is likely to be located close to a decision boundary. Brust et al. do not use localization when estimating image informativity, which decreases the potential benefits of the active learning process.

The third related work we want to introduce is by Hausmann et al. 2020 [14]. While the two previously introduced pieces of research used Pascal VOC or Coco datasets and used less than 1,100 images, Hausmann et al. used a dataset containing 2 million images recorded by the cars on the road. Their dataset is a private dataset from Nvidia, and the authors are Nvidia employees. In their research, several active learning methods were tested, and the best results were achieved when the predicted object classification confidences were used to calculate entropy, then top-N images were selected, and finally, possible similar images were filtered by using cosine similarity of the image embeddings. Hausmann et al. do not use localization when estimating image informativity, which, similarly to the research by Brust et al. [6], decreases the potential benefits of the active learning process.

1.3 Structure of thesis

Chapter 2 will present the standard building blocks used in the modern convolutional neural networks (CNN) object detection algorithms. We will not cover CNN layer types on details, but this information can be found from the Goodfellow et al. 2016 Deep Learning book [12]. After introducing the object detector architecture, we will present the common image augmentation techniques used in object detection and also in our experiment. These augmentation techniques include strong augmentations Mosaic [5] and mixup [44].

Chapter 3 covers the metrics used to evaluate object detection algorithms. These metrics can be defined in several different ways, and we follow the definition used in the Pascal VOC object detection challenge 2010 onwards. The most important object evaluation metrics in our experiment are Average Precision (AP) and Mean Average

Precision (mAP).

Chapter 4 introduces the YOLOX object detection algorithm. YOLOX is the latest YOLO family object detector, and it won CVPR 2021 streaming perception challenge [11][45]. First, we will go through the development history of the YOLO algorithms, and then we will describe the YOLOX algorithm and compare its performance to the other object detectors on COCO 2017 object detection dataset.

Then in chapter 5, we will go through the active learning concept. We will begin the chapter by comparing active learning to passive learning, and then we will introduce the standard active learning process. The active learning process has three main phases: defining seed set, most informative unlabelled data selection, and terminating the active learning process. We will introduce all these phases, while our research focuses only on the second phase, where we select the most informative images. At the end of chapter 5, we will describe how we can make the object detector training data annotation process more efficient by using active learning as annotation support.

In chapter 6, we will introduce our experiment where we leverage the object detector localization and classification confidences to sample the most informative images. In this chapter, we will introduce the experiment in detail, and at the end, we report the results and benefits that can be achieved by using our proposed method.

2. Convolutional Neural networks in Object Detection

Object detection is the task of localizing objects with tight bounding boxes and classifying their content [4]. On the typical task, object classes include people, cars, dogs, cats, etc., but the classes may be almost anything. In the field of object detection, the most famous datasets are Coco and Pascal VOC, where Coco has 80 different object classes, and Pascal 20 [25][8].

In figure 2.1, we demonstrate the object detection task. It also illustrates some of the challenges of high-class variability. The causes of the variability can be divided into two subgroups: variations due to the imaging process and the intrinsic appearance variability of objects [47].

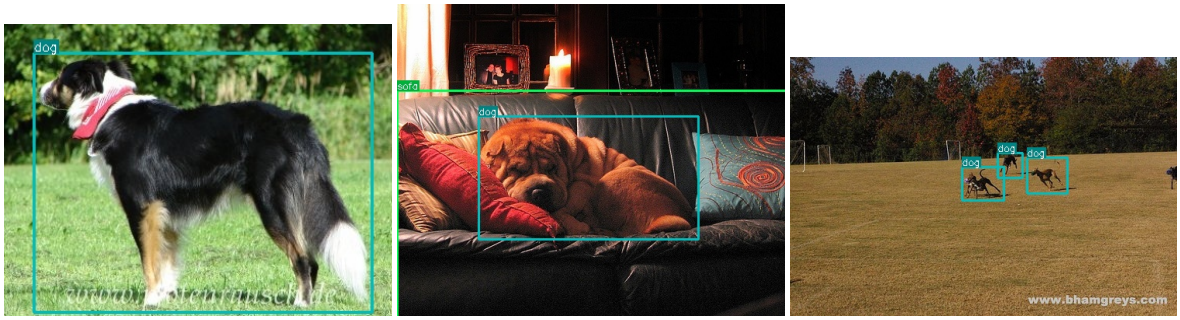

Figure 2.1: Three dog images from the Pascal VOC 2007 test set [8]. The name of the object class is presented on the left-top corner of the bounding box.

The variability that the imaging process causes are due to changes in the illumination, camera position, and artifacts caused by the camera, like motion blur [47]. We can effectively limit their impact on the model performance with the training data augmentation, a technique described in chapter 2.2. Then the intrinsic appearance variability is caused by the fact that even objects belonging to the same class may look very different. Training a robust CNN model against this type of variance requires a high-quality training dataset, but we can limit its impact with augmentation methods.

2.1 Object Detector Architecture

The majority of the modern-day CNN-based object detection algorithms use an architecture that has three parts. These parts are the backbone, neck, and prediction head, and the structure is visualized in figure 2.2 [5].

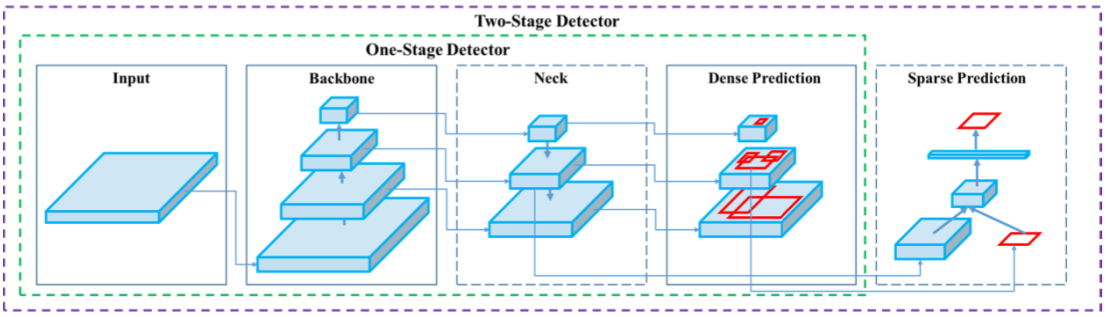

Figure 2.2: Typical the object detector architecture. A dense prediction head is used with the single-stage object detectors and a sparse head with the two-stage object detectors [5].

Since the Faster Region Based Convolutional Neural Networks (Faster R-CNN) [35] and the first version of You Only Look Once (YOLOv1) [32] object detectors, significant improvements have been made on improving these basic building blocks. Since the main focus of our paper is on the YOLO family object detectors, we focus to the innovations used in those. Next, we will go through these three object detector building blocks and their development during the past years.

The first part of the object detector is the backbone, which is used to extract features from the input images [18]. A typical backbone uses the structure of the CNN classification network, and we will use Darknet19, from the second YOLO version (YOLOv2) [33], as an example to describe an object detector backbone at a detailed level. The whole Darknet19 architecture is presented in figure 2.3.

The Darknet19 uses a block structure with 3x3 and 1x1 convolutional layers and at the end of each block is a pooling layer. When moving deeper, the output dimensions decrease, and the number of channels (filters) is doubled after each pooling step.

The details of the used layer types can be found in the Deep Learning book by Goodfellow et al [12]. As a summary, the purpose of the used layer types can be summarized as follow:

- 3x3 convolution is used to extract features [12].
- 1x1 convolution is used for cross-channel parametric pooling [22].
- The pooling layer is used for in-channel parametric pooling [12].

This process also changes the dimensions of the feature map, which is created from the input image in the first convolutional layer. As an example, let's consider a feature map with dimensions (h, w, and d), where h is height, w is width, and d is depth. With 1x1 convolution, we can manipulate the size of the d, and with the pooling layer, we can manipulate sizes of the h and w. This process can also be seen in figure 2.3.

As a total, Darknet19 has 19 convolutional layers and 6 pooling layers. Newer backbones, like Darknet53, also use residual shortcut connections [16] to create deeper and more powerful backbones [34].

Type	Filters	Size	Stride	Output dimensions
Convolutional	32	3 x 3	1	224 x 224 x 32
Maxpool		2 x 2	2	112 x 112 x 32
Convolutional	64	3 x 3	1	112 x 112 x 64
Maxpool		2 x 2	2	56 x 56 x 64
Convolutional	128	3 x 3	1	56 x 56 x 128
Convolutional	64	1 x 1	1	56 x 56 x 64
Convolutional	128	3 x 3	1	56 x 56 x 128
Maxpool		2 x 2	2	28 x 28 x 128
Convolutional	256	3 x 3	1	28 x 28 x 256
Convolutional	128	1 x 1	1	28 x 28 x 128
Convolutional	256	3 x 3	1	28 x 28 x 256
Maxpool		2 x 2	2	14 x 14 x 256
Convolutional	512	3 x 3	1	14 x 14 x 512
Convolutional	256	1 x 1	1	14 x 14 x 256
Convolutional	512	3 x 3	1	14 x 14 x 512
Convolutional	256	1 x 1	1	14 x 14 x 256
Convolutional	512	3 x 3	1	14 x 14 x 512
Maxpool		2 x 2	2	7 x 7 x 512
Convolutional	1024	3 x 3	1	7 x 7 x 1024
Convolutional	512	1 x 1	1	7 x 7 x 512
Convolutional	1024	3 x 3	1	7 x 7 x 1024
Convolutional	512	1 x 1	1	7 x 7 x 512
Convolutional	1024	3 x 3	1	7 x 7 x 1024
Convolutional	1000	1 x 1	1	7 x 7 x 1000
Avgpool		Global		1 x 1000
Softmax				1 x 1000

Figure 2.3: Darknet-19 classification model architecture [33].

The second part of the modern object detector is the neck, and it serves two main purposes. First, it improves object detectors' scale invariance by processing images in multiple scales. Second, it combines semantically strong low-resolution features with semantically weak high-resolution features [47]. The second benefit is achieved by combining feature maps from multiple feature levels [26].

It is possible to enhance the neck by adding an additional block [5]. In the newer YOLO architectures, this means using of spatial pyramid pooling (SPP) block that increases the receptive field and pools arbitrary size feature maps into fixed-size vectors. Because of this, we do not need to fix the input image size, and also model becomes more robust for detecting the objects in multiple scales [15].

In the YOLO context, the YOLOv1-2 [32][33] did not use any neck. Later, YOLOv3 [34] introduced neck with the Feature Pyramid Network (FPN) and [23] Spatial Pyramid Pooling (SPP) [15] blocks. In the YOLOv4 [5], FPN was replaced with a more advanced Path aggregation Network (PANet) [26]. Later YOLO models continue to use the same neck with the PANet and SPP [19][11].

The last part of the object detector is the head [5]. Head is where the object localization and classification happen, and several head architectures have been developed over time. The two main branches are the dense- and sparse prediction heads, also known as single and two-stage detection heads. In principle, the one-stage head simultaneously predicts multiple bounding boxes and class probabilities for those boxes [32]. This differs from the two-stage detection heads, where the head first predicts the bounding boxes, and then the content of the predicted bounding boxes is classified [35].

In the detection head, there are two different strategies for object localization. The first one is based on the predefined anchor boxes, and the second is anchor-free. The anchor-based localization was proposed by Ren et al. in the Faster R-CNN paper in 2015 [35], and the majority of the object detectors follow this strategy. In anchor-based localization, k predefined reference boxes are used to estimate if location l has an object. After the object has been found, reference boxes are combined to get the final bounding box. Faster R-CNN uses reference boxes with three different aspect ratios and three different scales, yielding $k = 9$. In 2019 anchor free object detector proposed by Tian et al. [40] overcame the performance of anchor-based object detectors. In anchor-free localization, object location is predicted instead of using predefined reference boxes. As it does not use predefined anchors, it requires less hand-made parameter tuning, and also, the prediction head gets simpler. In YOLOX, anchor-free localization was first time implemented in the YOLO architecture [11].

The object detector backbone, neck, and head can be summarized as [5]:

1. Backbone:
   • Darknet19 [33] • Darknet53 [34] • CSPDarknet53 [5] • ResNet-101 [16]
2. Neck:
   Path-aggregation blocks: • FPN [23] • PANet [26]
   Additional blocks: • SPP [15]
3. Head:
   Dense Prediction (one-stage):
   Anchor based: • YOLOv1-YOLOv5 heads [32][33][34][5][19]
   Anchor free: • YOLOX head [11] • FCOS head [40]
   Sparse Prediction (two-stage):
   Anchor based: • Faster R-CNN [35]

2.2 Data augmentation

Data augmentation is an important part of training deep convolutional neural network-based object detectors, and its purpose is to enhance the size and quality of training datasets. In practice, this means a process of artificially creating new images by applying photometric and geometric distortions to training images [5]. Then by using the original and augmented images, we can train the object detection model that is more robust against the variations in environmental conditions, variations in imaging processing, and variations in intrinsic appearance [37].

Next, we will introduce the data augmentation techniques used in the standard YOLOv4, YOLOv5, and YOLOX training processes. These techniques are hue, saturation, value (HSV) manipulation, horizontal flipping, rotation, translation, shearing, scaling, mosaic, and mixup. Two example images with these augmentations can be found in figure 2.6.

The first augmentation technique is the HSV color channel manipulation. In this process, image hue, saturation, and value are randomly manipulated inside predefined limitations [11]. The trained model will be more robust against the lightning and color variation in the images by using HSV augmentation. The second augmentation is random horizontal image flipping. In this process, images with probability p get horizontally flipped. As a result, the trained model will produce the same output if the object is located left-to-right or right-to-left. It is not common to use vertical flipping in object detection, as it is not expected that objects could appear in the images upside-down. The third augmentation is rotation. In the case of YOLOX, image rotation is a random angle between -10 to 10 degrees. With image rotation augmentation, we train

the model that may appear in the image at different angles. The fourth augmentation technique is image translation. This means a process of moving the image along with the X and Y directions. As a result, we train the model that the object may appear at any location in the image. The fifth augmentation is the shear transformation. In shear transformation, we fix one image axis and then rotate the rest of the image according to the random shearing angle. With shear augmentation, we create training images as they were taken from different directions. The sixth basic augmentation is scaling. As the name suggests, in scaling, we change the image size.

These six techniques belong to the basic augmentation techniques, and example images where these are applied are presented in figure 2.4. Next, we will present so-called strong data augmentation techniques Mosaic and mixup.

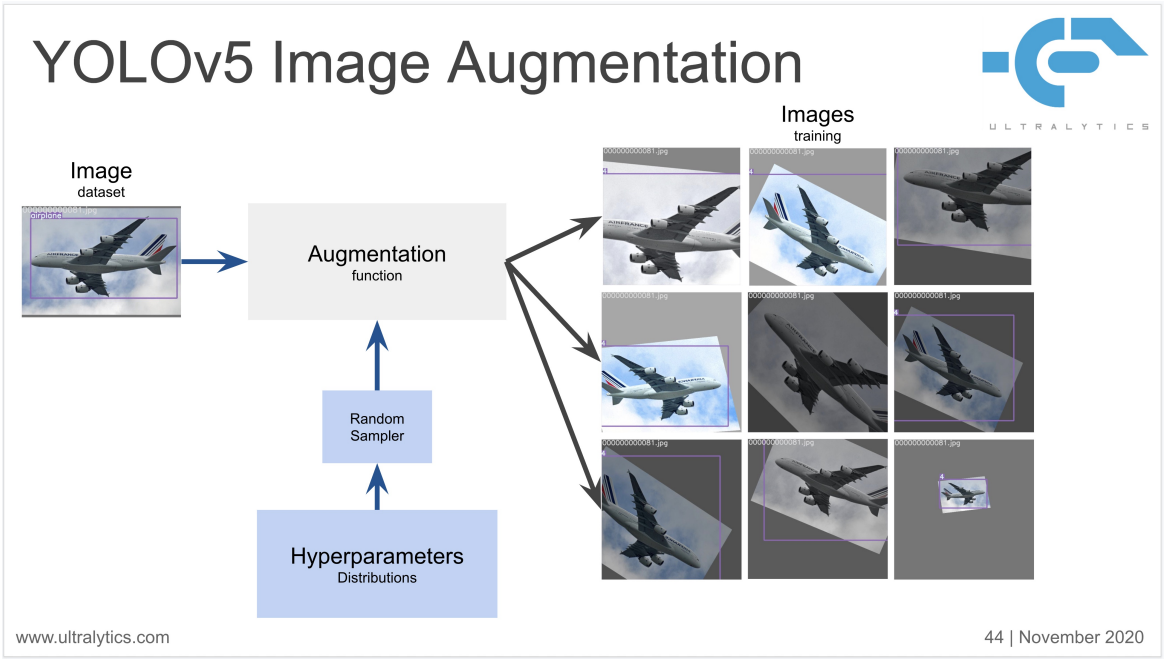

Figure 2.4: Example images with data augmentations: HSV manipulation, horizontal flipping, rotation, translation, shearing, and scaling [19].

The Mosaic and mixup augmentation belong to the so-called strong data augmentation techniques. Mosaic augmentation follows the principles of CutMix [43] augmentation, where patches from multiple images are cut out and then pasted together. In CutMix, two patches were used, but YOLOv4 authors [5] found out that using four patches would be more efficient, and this technique is called Mosaic augmentation. Mosaic augmentation allows training object detection algorithms to detect objects outside their normal context, and by having multiple images in one image, a smaller mini-batch size can be used. Figure 2.5 presents example images where only Mosaic augmentation has been applied.

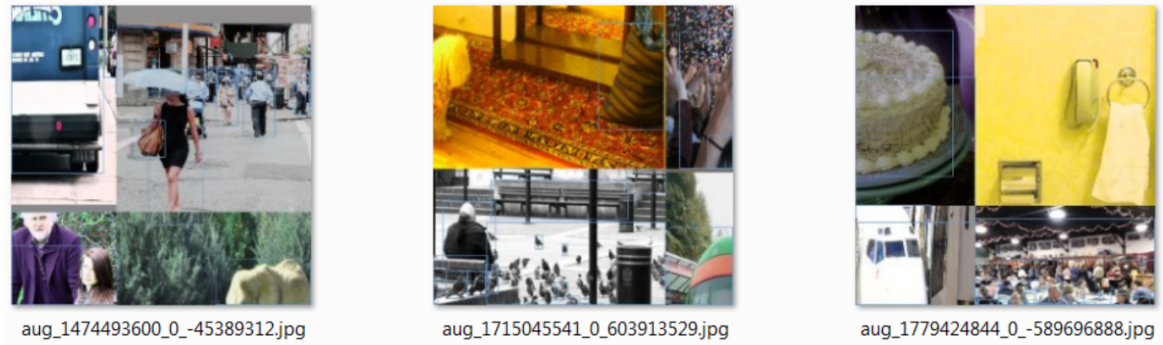

Figure 2.5: Mosaic augmentation examples [5].

The last augmentation technique is the mixup [44]. In mixup, training images i and j are placed on top of each other with the magnitude of $\lambda \in [0, 1]$. We define mixup as

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j \tag{2.1}$$

$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j \tag{2.2}$$

where x_i and x_j are raw input vectors, y_i and y_j are one-hot label encodings, $\tilde{x}$ is the new combined image, and $\tilde{y}$ the combined label.

Mixup is based on the prior knowledge that linear interpolations of the features should lead to the interpolation of the associated targets. As a result, mixup trains the CNN model to favor linear behavior in-between the training examples [44].

After we apply all nine augmentation techniques presented in this chapter, we get training images, as shown in figure 2.6. Combining presented augmentation methods have been proved to be an effective way to improve output object detector performance [11], which is the reason we use these augmentations in our experiment in chapter 6. By using these augmentations, our active learning experiment follows the techniques that would also be used in the real-life use case [45].

2.3 Real-time and slower object detectors

With the state of the art object detection models, there is always a trade-off between the model performance and the inference speed. This is not very surprising as the deeper networks have more capacity to learn more details. Also, it is common practice to downscale image resolution before inference to make the inference faster, and it is common to use 640x640 image size as the model input. Downside the downscaling

Figure 2.6: Two training images with all nine data augmentations: HSV manipulation, horizontal flipping, rotation, translation, shearing, scaling, Mosaic, and mixup [19].

image resolution is the lost information, but as said before, there is always a trade-off between speed and accuracy [45]. Figure 2.7 presents the YOLOX large version (YOLOX-L) performance-to-speed comparison with the different input image sizes.

Image width	Image height	Speed (ms)	AP
1440	2304	28.1	50.6
1280	2048	21.4	49.9
1200	1920	20.5	49.7
1120	1792	19.7	48.7
960	1536	16.0	46.3

Figure 2.7: Image size vs. Speed vs. Precision with YOLOX-L object detector [45].

Generally, the object detectors can be divided into two groups based on the inference speed. The first group is the real-time models, where the common speed requirement is 30 frames per second (FPS) on a single GPU. In the real-time models, the YOLO models have been the most popular ones and it has been updated multiple times with the latest techniques [5][11]. Commonly the real-time models are single-stage detectors, while there is more variance among the slower and more powerful models.

The second group can be called as not real-time object detectors. The most popular family in this group has been the R-CNN model family, which was later updated to the Fast R-CNN model, then to Faster R-CNN model [35], and later the

Faster R-CNN model with feature pyramid network (FPN) [23]. This model family belongs to the two-stage object detectors, where the first stage localizes the object and the second stage classifies it, while in the single-stage detectors, both of these happen simultaneously.

Since 2020, transformers-based object detectors are starting to show promising results [7]. They are still significantly slower than the single-stage CNNs, but they are powerful and fast enough to compete against the slower models. Also, transformer encoder-decoder structures simplify the overall model architecture and minimises the need for the hand-crafter rules, like the commonly used anchors [49].

3. Performance Metrics for Object-Detection Algorithms

A wide range of evaluation metrics has been developed to evaluate the performance of the object detection models. It is challenging to summarize the model performance with a single number in object detection due to the two subtasks: localization and classification. Due to this challenge and the lack of common consensus of the used metric, evaluating the object detection models is a challenge faced by the scientific community [31].

The commonly used metrics include average precision (AP), the area under the curve (AUC), precision, recall, average recall (AR), average precision at IoU greater than 0.50 (AP50), and many others. Besides the wide range of different metrics, many of the metrics can be defined in multiple ways. One example is average precision, which can be calculated in at least six different ways [31].

The next chapters will introduce the metrics used in our research and explain how they are calculated. We use the metrics and evaluation as they are defined in the Pascal VOC object detection challenges from 2010 onwards [8].

3.1 Definition of correct localization

Before defining the most important metric, average precision, we need to define background formulas. First, we will define the meaning of correct localization.

In object detection, the object location areas are called bounding boxes. Bounding boxes are commonly defined as the coordinate of the left-top corner and the coordinate of the right-bottom corner. Another common way is to define the object center point and then the object width and height. In both cases, the ground-truth and the object detection model's localization result are defined with the precision of a single-pixel [31].

However, we want to accept a small error in the localization due to two reasons. First, the human annotator draws the ground truth locations, which may have minor errors. Secondly, it is common enough to have an estimated object location, and in fact, humans are poor at visually estimating the localization tightnesses [36].

The standard way to define correct localization without requiring exactly the same result is the Intersection Over Union (IoU). IoU is calculated as a Jaccard Index J, which measures a similarity coefficient between two bounding boxes [31]. IoU of the predicted bounding box B_p and the ground truth bounding box B_{gt} is defined as:

$$IoU = J(B_p, B_{gt}) = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}, \tag{3.1}$$

where $B_p \cap B_{gt}$ denotes the intersection of the predicted and ground truth bounding boxes and $B_p \cup B_{gt}$ denotes their union [8].

In figure 3.1 we illustrate the IoU calculation with an example. In the numerator, we calculate the intersection areas, and in the nominator, we calculate the union of the areas. As a result, we get the IoU of these two areas.

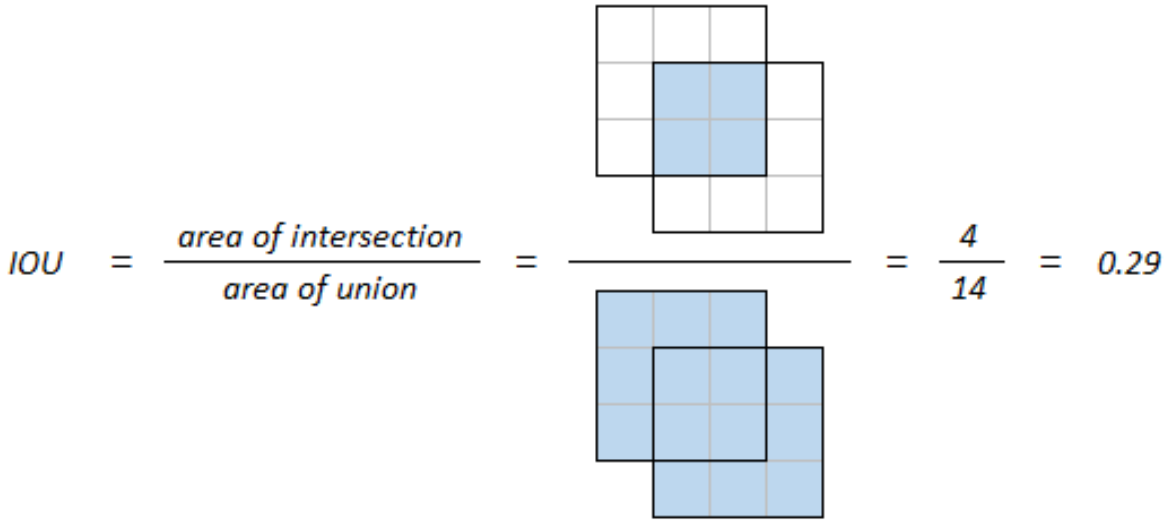

Figure 3.1: Example of Intersection Over Union (IoU).

We also want to highlight how different IoU levels look in practice, as humans are naturally poor at visually evaluating IoU [36]. In figure 3.2, we have two images with increasing IoU values. These images show that localization tightness looks surprisingly good already on lower IoU values, and based on this observation, we can justify the decision to accept IoU values less than 1.0.

With the IoU we can flexibly define if the localization is correct or not. As we want to accept IoU values less than 1.0, we use lower-level threshold t, where $t \in [0, 1]$, and bounding boxes with $IOU \geq t$ are considered correct localization (True Positive). Similarly, if $IoU < t$, the localization is considered incorrect (False Positive). In practice, it is common to use $t = 0.5$ or $t = 0.75$ [31].

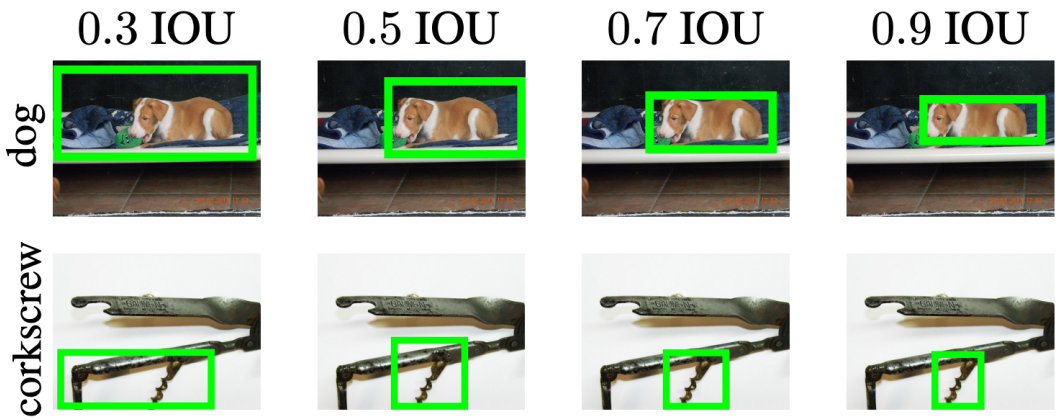

Figure 3.2: Two example images with increasing intersection over union (IoU) [36].

3.2 Precision and Recall

So far, we have defined the meaning of the correct localization. Next, we need to extend the definition to define precision and recall, which are used to evaluate model performance when detecting multiple objects from several object classes.

When we evaluate an object detection model, the outputs can be divided into three groups. These are:

- True Positive (TP): Localization and classification correct
- False Positive (FP): Detection of a nonexistent object or localization IoU less than threshold t (classification correctness is irrelevant).
- False Negative (FN): Undetected ground-truth bounding box (no classification result, as the object is not detected).

The true negative (TN) is not used in object detection, as there are infinite number of possible bounding boxes that should not be detected in a single image [31]. Therefore, we need to use metrics that do not require information about the true negatives. Precision P and recall R are such metrics, which is why the evaluation of object detection models is based on these two concepts.

Precision and recall are defined as:

$$Precision\ (P) = \frac{TP}{TP + FP} = \frac{TP}{all\ detection} \tag{3.2}$$

$$Recall\ (R) = \frac{TP}{TP + FN} = \frac{TP}{all\ ground\ truths} \tag{3.3}$$

The precision metric measures the models' ability to detect and classify only relevant objects. In other words, it penalizes the model for possible false positive detections. Then the recall measures the models' ability to detect and classify all the relevant objects, also known as ground truth bounding boxes [31]. In summary, precision penalizes the false positive detections, and recall penalizes for not detecting all the relevant objects.

As precision and recall measure different properties, they are both important in evaluating the object detecting model's performance. An object detector with only a high recall or high precision would work poorly in practice. Instead, a good object detector requires both high precision and high recall [31].

3.3 Average Precision (AP) and Mean Average Precision (mAP)

The average precision (AP) and mean average precision (mAP) are the most important metrics in object detection. AP is used to evaluate object detections in a single object class, and mAP is used to evaluate object detectors' overall performance. These metrics can be defined in several different ways, and next, we will introduce the definition used in the Pascal VOC object detection challenge from 2010 onwards [8]. This is also the definition we follow in our research in chapter 6.

The AP is defined using the area under the curve (AUC). As a good model should have both high precision and high recall, we want to maximize AUC in the precision/recall plot.

To create the precision/recall plot, we first gather all the detector results for a wanted object class, and as a result, we get a table presented in figure 3.3. In figure 3.3, detections are ordered by the detection confidence. We want to make the precision/recall curve monotonically decreasing, meaning that when recall increases, then precision decreases. That's why we will interpolate precision values and select the highest precision value for each recall value [9]. The formula to calculate interpolated precision values P_{interp} is defined as:

$$P_{interp}\left(R\right) = \max_{\tilde{R}:\tilde{R}\geq R} P(\tilde{R}), \tag{3.4}$$

where P_{interp} is maximum precision value P, with recall value $\tilde{R}$ greater or equal to R [31]. In figure 3.3, we present this calculation with toy problem detections and visualizes the area under curve plot based on the table precision, interpolated precision, and recall values.

Confidence	TP	FP	Acc TP	Acc FP	P	R	P_{interp}
95 %	1	0	1	0	**1**	0.0666	1
95 %	0	1	1	1	0.5	0.0666	1
91 %	1	0	2	1	**0.6666**	0.1333	0.6666
88 %	0	1	2	2	0.5	0.1333	0.6666
84 %	0	1	2	3	0.4	0.1333	0.6666
80 %	0	1	2	4	0.3333	0.1333	0.6666
78 %	0	1	2	5	0.2857	0.1333	0.6666
74 %	0	1	2	6	0.25	0.1333	0.6666
71 %	0	1	2	7	0.2222	0.1333	0.6666
70 %	1	0	3	7	0.3	0.2	0.4285
67 %	0	1	3	8	0.2727	0.2	0.4285
62 %	1	0	4	8	0.3333	0.2666	0.4285
54 %	1	0	5	8	0.3846	0.3333	0.4285
48 %	1	0	6	8	**0.4285**	0.4	0.4285
45 %	0	1	6	9	0.4	0.4	0.4285
45 %	0	1	6	10	0.375	0.4	0.4285
44 %	0	1	6	11	0.3529	0.4	0.4285
44 %	0	1	6	12	0.3333	0.4	0.4285
43 %	0	1	6	13	0.3157	0.4	0.4285
38 %	0	1	6	14	0.3	0.4	0.4285
35 %	0	1	6	15	0.2857	0.4	0.4285
23 %	0	1	6	16	0.2727	0.4	0.4285
18 %	1	0	7	16	**0.3043**	0.4666	0.3043
14 %	0	1	7	17	0.2916	0.4666	0.3043

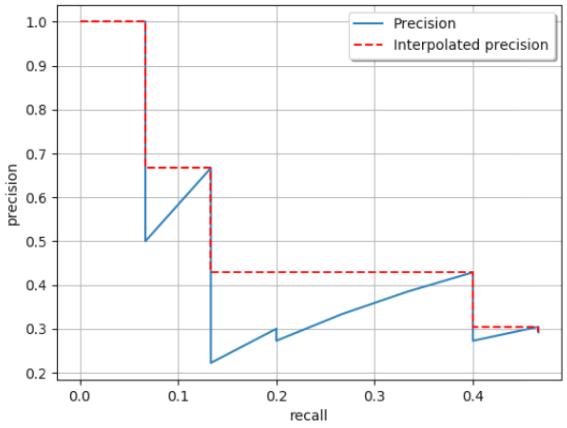

Figure 3.3: Precision and recall example. The left side table contains 24 detections for the single object class, where 7 are true positives, and 17 are false positives. The table values are visualized on the right-side plot as a precision/recall curve. Acc TP is an accumulative count of True Positives, and Acc FP is an accumulative count of False Positives. In the table, maximum precision values used in the P_{interp} are bolded. Dashed horizontal lines are used to highlight the P_{interp} values [31].

Then we can calculate the AP. AP is the area under interpolated precision to recall curve, and it is defined as:

$$AP = \sum_{n}(R_{n+1} - R_n)P_{interp}(R). \tag{3.5}$$

This is the AP definition we use in our experiment. In the experiment, we use the notation of AP50, which means AP with IoU$\geq$ 0.5.

After calculating the AP values for each object class, we can calculate the mean average precision (mAP). The mAP is used to summarize per class average precision (AP) values into a single value. The formula to calculate mAP is defined as:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i, \tag{3.6}$$

where AP_i is AP value of the ith object class and N is the total number of object classes.

4. YOLOX

You Only Look Once (YOLO) series object detectors [32] [33] [34] [5] [19] [11] are popular real-time deep learning-based object detectors. The YOLO series detectors have always looked for the optimal speed and accuracy trade-off for real-time applications.

The YOLO detectors have been frequently updated with the latest object detection techniques. As a result, the YOLO detectors' performance has been steadily increasing through the versions without sacrificing the inference speed, and they have become popular in many real-life applications. Also, the latest YOLO detector, YOLOX, is used in the state-of-the-art (SOTA) real-time multi-object tracking network [46].

The first YOLO series object detector was published in 2016 [32], and since that, it has been improved several times. The original research group developed the first three versions, which after other research groups have released their own and updated versions of the YOLOv3. The whole YOLO detectors development timeline is introduced in figure 4.1, and we introduce the YOLO detectors history in chapter 4.1.

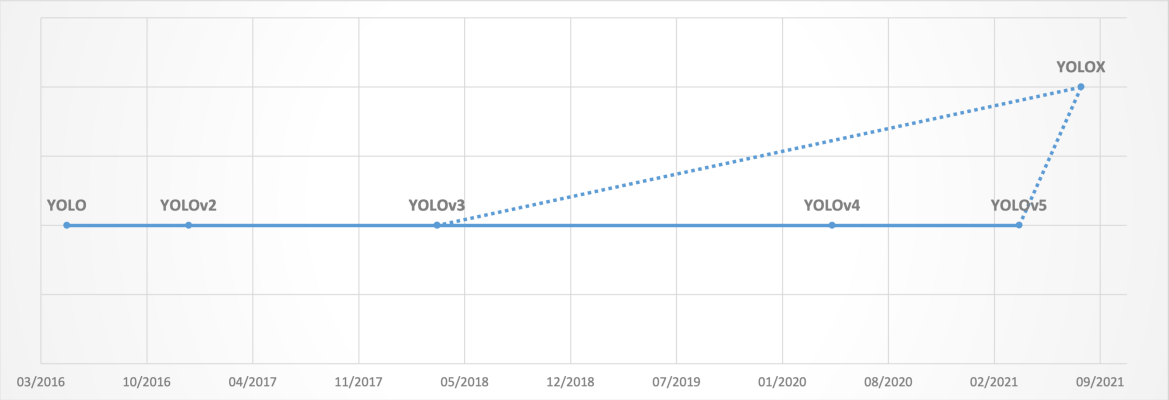

Figure 4.1: YOLO development timeline.YOLOX is based on the YOLOv3, but YOLOX versions YOLOX-S, YOLOX-M, YOLOX-L, and YOLOX-X use modified CSPNet as their backbone. Modified CSPNet was first introduced on the YOLOv5. For this reason, there is a connection from YOLOX to both the YOLOv3 and YOLOv5.

The latest and the most sophisticated YOLO detector is called YOLOX. YOLOX was developed by Ge et al. [11] from Megvii Technology, and it was published in August 2021. According to the developers of the YOLOX, they wanted to implement the latest

object detection techniques to the YOLO architecture, and as a result, became YOLOX. As a starting point, Ge et al. used YOLOv3 because the later YOLO architectures were over-optimized for the new techniques [11]. YOLOX has a connection to the YOLOv5 by using the same backbone.

When compared to the older YOLO versions, YOLOX uses multiple new techniques, which as a result, gives an impressive speed and performance. The YOLOX has also won the 2021 CVPR workshop on the autonomous driving streaming perception challenge, proving it's a State-of-the-Art (SOTA) model [11].

4.1 YOLO detectors history

The first version of the YOLO (You Look Only Once) was released in May 2016 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. YOLO was developed in cooperation between the University of Washington, Allen Institute for AI, and Facebook AI Research [32]. The first YOLO version was a significant step towards accurate real-time deep learning object detectors, and since its release, until 19th of January 2022, it has been cited 21,369 times.

When YOLO was released, it offered a completely new approach to object detection. The theory of the YOLO is to frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. Before YOLO, the solution to the object detection problems was to repurpose classifiers to perform detection [32]. As a result, YOLO offered significant speed improvement while the performance was close to the most accurate object detection models at that time. A comparison between the YOLO and other object detectors from that time is presented in figure 4.2.

According to the YOLO authors, the comparison presented in figure 4.2 does not tell the whole truth. When compared to the state-of-the-art detection systems, YOLO makes more localization errors, but at the same time, it is less likely to predict false positives in the background [32].

The YOLO object detection process has four steps, where the second and third steps run in parallel. The process is also described in the image 4.3. In the first step, the image is divided into the S x S grids. Then in the second step, the model predicts B bounding boxes (x, y, w, h) and object confidence for each bounding box. Object confidence is the confidence of whether the bounding box has an object or not. In the parallel is running the third step, where the model runs classification on each grid cell. In the last step, the results of the localization and classification results are combined. Boxes with high object confidence are assigned grid classification results, which give the object location (x, y, w, h) and the object's class in the given grid cell. Then we

Method	Real-time	Train	mAP	FPS
Fast YOLO	Yes	2007+2012	52.7	**155**
100Hz DPM	Yes	2007	16.0	100
YOLO	Yes	2007+2012	63.4	45
30Hz DPM	Yes	2007	26.1	30
YOLO VGG-16	No	2007+2012	66.4	21
Faster R-CNN ZF	No	2007+2012	62.1	18
Fastest DPM	No	2007	30.4	15
Faster R-CNN VGG-16	No	2007+2012	**73.2**	7
R-CNN Minus R	No	2007	53.5	6
Fast R-CNN	No	2007+2012	70.0	0.5

Figure 4.2: Speed and accuracy comparison on PASCAL VOC2017 dataset. Speed measured in Titan X GPU and list is ordered by FPS [32].

run non-maximal suppression to join multiple boxes and remove overlapping boxes of the same class object [32].

In summary, the YOLO architecture is simpler and faster than other high-accuracy object detection architectures. Because YOLO does not use sliding windows, YOLO sees the entire image and encodes contextual information and the appearance of the classes. As a result, YOLO is highly generalizable, but it has some difficulties in precisely localizing some objects. Also, as YOLO runs classification to grid cells, it is not able to classify multiple nearby objects that might belong to different classes [32].

Seven months after the release of the first version, the original authors released the second YOLO version. The second version is officially named YOLO9000 [33], but later it was called YOLOv2 to highlight that it is the second YOLO version. The name of the YOLO9000 comes from its capability to localize and classify over 9000 different objects while running in real-time.

The second YOLO version improved the two main weaknesses the first version had. The first YOLO version made a significant number of localization errors, and it had lower recall compared to the region proposal-based object detectors. These were the main objectives YOLOv2 developers wanted to tackle while also maintaining high classification accuracy and fast inference speed [33].

The developers did not want to compromise the speed, so using a deeper network was not an option. Instead, they implemented four new techniques on the YOLOv2 [33]. These four new techniques were:

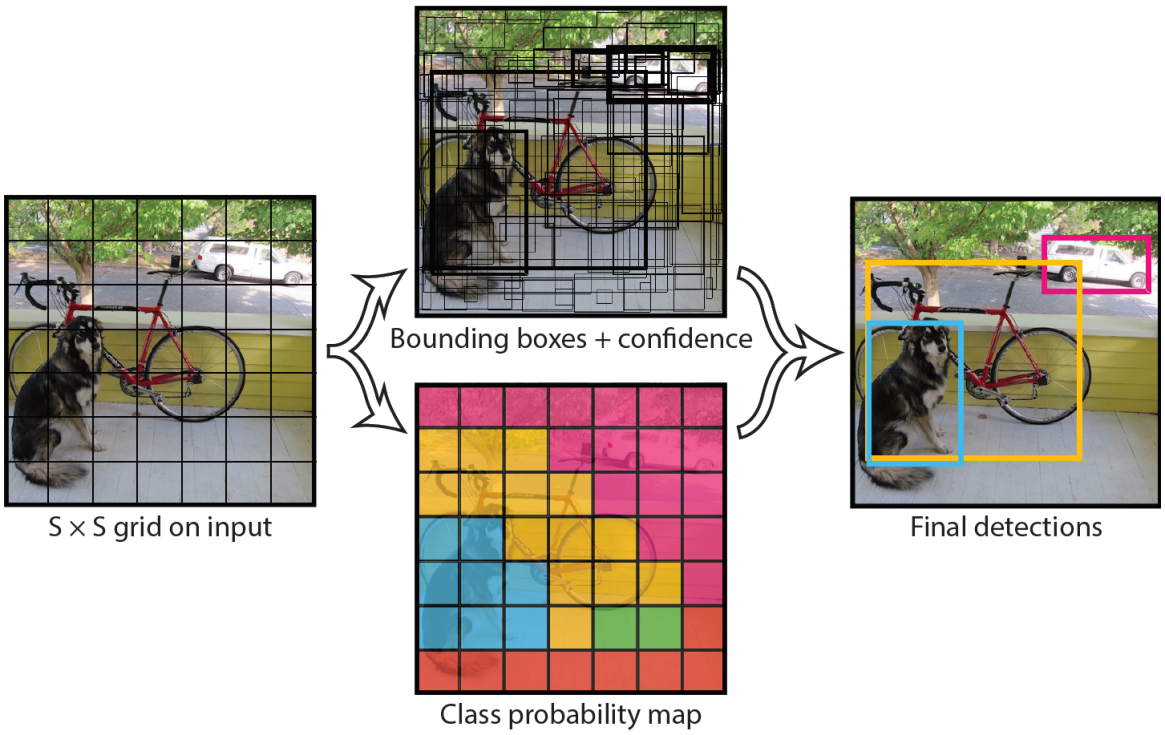

Figure 4.3: YOLOv1 object detection process [32].

1. Batch normalization
2. Anchor boxes
3. Fine-Grained Features
4. New classifier, Darknet-19

The first new technique was batch normalization [17]. Batch normalization improves model convergence during the training and helps to regularize the model. As a result, it was possible to remove dropout and still get a 2 % better mAP score than the YOLOv1 [33].

The second significant change was starting to use anchor boxes to predict bounding boxes similarly as in the Faster R-CNN [33][35]. Using anchor boxes in the YOLO network worsens the mAP score slightly, but increases improve the recall from 81 % to 88 %. In the YOLOv1, there were 98 boxes, but with the anchor boxes, the number of boxes increased to thousands. In the YOLOv2, the classification network was changed to classify each anchor box instead of grids as done in the YOLOv1. Even the anchor boxes decrease the mAP score, the improved recall means that the model has more potential to improve [33].

The third change was to use fine-grained features, which follow the idea of the ResNet [16]. In the YOLOv2, this means that high-resolution features are concatenated

with low-resolution features by stacking them into different channels. This process is part of the passthrough layer, and using both high- and low-resolution improves the localization precision. As a result, the mAP score was increased by 1 % [33].

The fourth change was a new classification model called Darknet-19 [33]. The Darknet-19 is faster and more accurate than the previously used classifier, and it is able to achieve 91.2 % top-5 accuracy on the ImageNet dataset. Darknet-19 was used as an example in chapter 2.1, where it was also explained at a detailed level.

These changes improved the YOLO mAP score from 63.4 to 78.6, making it only slightly slower. The speed dropped from 45 frames per second to 40 frames per second, which is still above the common real-time speed requirement of 30 frames per second. The improvements helped improved to detect and localize small objects, but it did not completely remove this flaw [33].

The third version of the YOLO detectors was released in 2018, slightly over a year after the YOLOv2. The YOLOv3 [34] was the last version developed by the original YOLO authors, and the later YOLO versions [5] [19] [11] heavily rely on the YOLOv3 network. At the end of the YOLOv3 research paper, the authors criticize computer vision in unethical applications, like military applications and personal data harvesting, which is the probable reason why YOLOv3 was their last version.

Interestingly, according to the authors, the motivation to publish the YOLOv3 paper was the following: "We have a camera-ready deadline [13] and we need to cite some of the random updates I made to YOLO but we don't have a source. So get ready for a TECH REPORT!" [34]. Even though it was released as a tech report, it is one of the most important research papers in object detection when measured by the number of citations.

The YOLOv3 introduced two significant improvements when compared to the previous YOLOv2. The first change was predictions across scales, similar to the feature pyramid network introduced two years before by Lin et al. [23]. The second major change was the new feature extractor, named Darknet-53 4.4.

The Feature Pyramid Network (FPN) introduced a computationally efficient and accurate way to make predictions in multiple scales, which tackles the issue where computer vision models have difficulties accurately detecting and classifying objects in various sizes [23]. While the previous YOLO models already delivered good accuracy with the medium and large size objects, they struggled to detect and localize small size objects [34]. Unlike previous YOLO models, YOLOv3 makes predictions at three different scales making localization and classification more accurate.

The second significant change was the new feature extractor. The new Darknet-53 is a hybrid between the Darknet-19, feature extractor used in the YOLOv2, and the residual learning framework introduced in 2016 by He et al. [16]. With the help of

the residual shortcut connection, the network is significantly deeper and more powerful than the Darknet-19. The number of convolutional layers has been increased to 53, which is also the logic behind the name. The Darknet-53 reaches similar accuracy as deeper ResNet-101 and ResNet-152 feature extractors while being significantly faster [34]. The whole Darknet-53 architecture is described in figure 4.4.

Block Count	Type	Filters	Size	Stride	Output
	Convolutional	32	3 x 3		256 x 256
	Convolutional	64	3 x 3	2	128 x 128
	Convolutional	32	1 x 1		
1x	Convolutional	64	3 x 3		
	Residual				128 x 128
	Convolutional	128	3 x 3	2	64 x 64
	Convolutional	64	1 x 1		
2x	Convolutional	128	3 x 3		
	Residual				64 x 64
	Convolutional	256	3 x 3	2	32 x 32
	Convolutional	128	1 x 1		
8x	Convolutional	256	3 x 3		
	Residual				32 x 32
	Convolutional	512	3 x 3	2	16 x 16
	Convolutional	256	1 x 1		
8x	Convolutional	512	3 x 3		
	Residual				16 x 16
	Convolutional	1024	3 x 3	2	8 x 8
	Convolutional	512	1 x 1		
4x	Convolutional	1024	3 x 3		
	Residual				8 x 8
	Avgpool		Global		
	Connected		1000		
	Softmax				

Figure 4.4: Darknet-53 classification model architecture [34].

Overall, YOLOv3 offers significant improvements over the previous YOLO models without sacrificing the network simplicity required to reach real-time inference speed.

After the YOLOv3, YOLO architecture stayed untouched for almost two years. In April 2020, new authors, Bochkovskiy et al. [5] released the fourth YOLO version, YOLOv4. According to the authors, researchers have developed a vast number of new techniques that are said to improve convolutional neural network accuracy. Still, practical testing of combinations and theoretical justification of the results are required. As a result of testing latest techniques became YOLOv4.

YOLOv4 made several architectural improvements to the YOLOv3 and added Mosaic and mixup data augmentation methods, which are introduced in chapter 2.2. YOLOv4 uses a new CSPDarknet53 backbone, SPP additional module, PANet path-aggregation neck, and YOLOv3 anchor-based head on the model architecture side [5].

The Cross Stage Partial Network (CSPNet) is a skip-connection technique that enables the backbone to achieve richer gradient combinations while reducing computation [5]. This is done by partitioning the feature map of the base layer into two parts. Then one part runs through the network stage, and at the end of the network stage, these two parts are merged. According to the CSPNet authors, the new skip-connection reduces computation by 20 %, resulting in equivalent or even better accuracy [41]. When Darknet-53 is updated with the CSPNet, we reach similar improvement, and the updated backbone is called CSPDarknet-53.

With the new, the CSPDarknet53 is added a Spatial Pyramid Pooling block (SPP) [5]. After the last convolutional layer, the SPP block does a max-pooling in multiple scales, significantly increasing the receptive field, meaning that far away features can be more efficiently and accurately utilized [15]. Also, as can be seen in figure 4.5, the SPP block pools arbitrary size feature maps into fixed-size vectors. Because of this, we do not need to fix the input image size, and YOLOv4 can be trained with several input image sizes, which makes YOLOv4 more robust for detecting the objects in multiple scales.

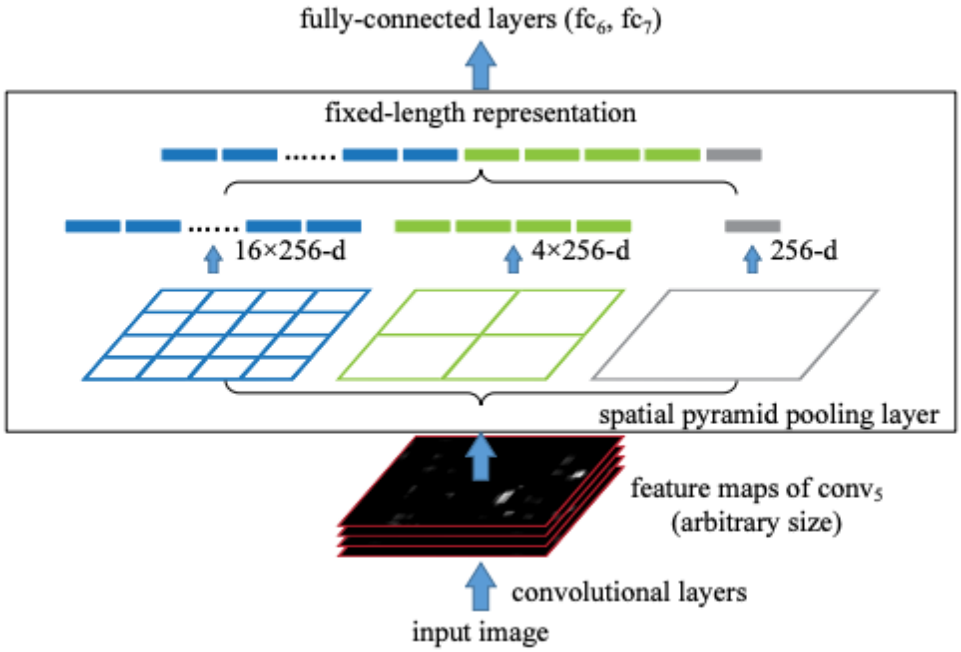

Figure 4.5: Spatial Pyramid Pooling (SPP) block. In image 256 is the number of filters in the last convolutional layer [15].

The last update YOLOv4 brought for the YOLOv3 is the Path Aggregation Network (PANet) method [5]. While the YOLOv3 used Feature Pyramid Network (FPN) in parameter aggregation from different backbone levels for the different feature levels, in YOLOv4, FPN is replaced with the PANnet. PANet shortens the information path between the low- and high-level features and, that way, improves localization accuracy [26].

The last version before the YOLOX is the YOLOv5 [19]. Authors of the YOLOv5 have not ever released paper from this object detector, which made it challenging to find the actual changes compared to the YOLOv4. The only change we found was the improved backbone called Modified CSP v5.

Overall, the Modified CSP v5 backbone is very similar to the CSPDarknet53, and the only difference is a small change in the number of blocks per stage [19].

Nevertheless, YOLOv5 brought one exciting update. Before YOLOv5, the backbone dimensions and layers were fixed. However, in YOLOv5, the convolutional kernel dimensions and block counts are parametrized. This means that the YOLOv5 can be easily scaled to different sizes. Authors of the YOLOv5 have named the main sizes as YOLOv5-S (small size), YOLOv5-M (medium size), YOLOv5-L (large size), YOLOv5-X (extra large size), where the first one is the smallest and fastest and the last one is largest and most accurate [19]. We can see how the architecture size affects the inference speed and accuracy from figure 4.8.

4.2 YOLOX - latest innovations to YOLO architecture

YOLOX updates the YOLO detectors with the latest academic innovations that can be applied to object detectors [11]. The YOLOX authors considered using YOLOv4 or YOLOv5 as the starting point for the new anchor-free object detectors, but these two were found to be over-optimized for the new anchor-free pipeline. As a result, the YOLOX authors decided to use YOLOv3 architecture as a starting point for the new anchor-free object detector while using the CSPNet backbone of the YOLOv5.

When we compare YOLOX to the previous YOLO models, it offers three important updates:

1. Anchor-free localization with IoU prediction

2. Decoupled head

3. SimOTA label assignment strategy

The most significant update in the YOLOX is to be the first anchor-free YOLO detector. Anchor-free localization means that the bounding box coordinates are predicted instead using predefined anchor boxes. By removing the anchor mechanism, overall architecture can be simplified, and the number of manually tunable parameters is significantly reduced [11]. With the anchor-free localization, an IoU prediction head is also added. The IoU predictions are used to get estimated localization confidences, which are used with the classification confidences to filter out possibly false positive predictions and to give overall detection confidence. In the YOLOX, the combined detection confidence (also called score) for single detection i (det_i) is defined as:

$$S_{total}(det_i) = S_{cls}(det_i) * S_{loc}(det_i), \tag{4.1}$$

where S_{cls} is the classification confidence of the most confident object class, and S_{loc} is the predicted IoU. Because the $S_{cls} \in [0, 1]$ and $IoU \in [0, 1]$, then the $S_{total} \in [0, 1]$. According to Kang et al., the predicted IoU naturally corresponds to the localization quality, and the range of the classification confidence matches with the predicted IoU, the intuitive way to combine these two scores is to use multiplication [40]. This is also the formula we use in our active learning research experiment in chapter 6 to calculate detected object confidences.

The second change is the new decoupled head, as introduced in figure 4.6. Like the anchor-free localization, the decoupled was also invented a few years before and used successfully in the other object detector architectures, like the RetinaNet [24]. Now, YOLOX brought this improvement to the YOLO architecture [11].

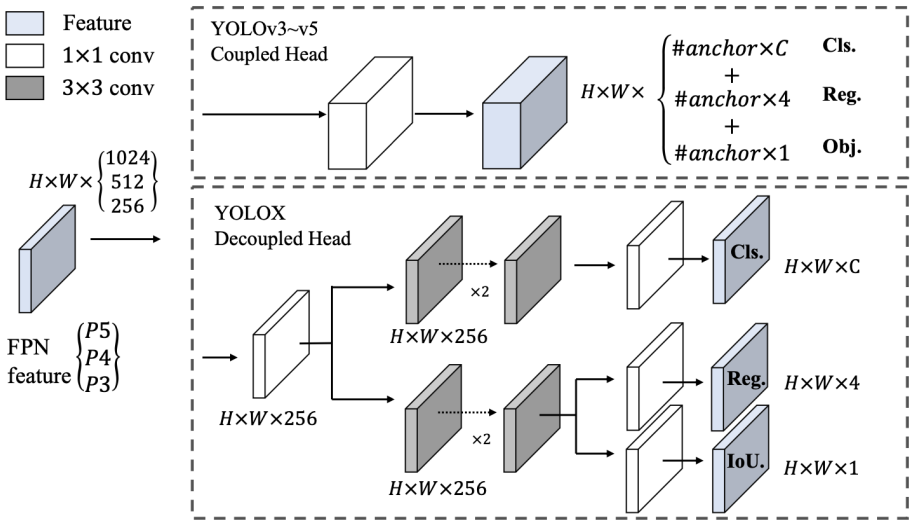

Figure 4.6: Illustration of the difference between the YOLOv3 to YOLOv5 coupled head and YOLOX decoupled head [11].

The benefit of the decoupled head comes from the fact that the object detector has two tasks. These are localizing and classifying the wanted objects in the images. However, these two tasks have spatial misalignment, and therefore the optimized joint head will always be a compromise [38]. Decoupled head removes the need to compromise between localization (regression) and classification. Because regression and classification tasks have their heads, they can be better optimized, improving model performance. Also, the decoupled head model converges significantly faster than the original YOLO head, as shown in image 4.7.

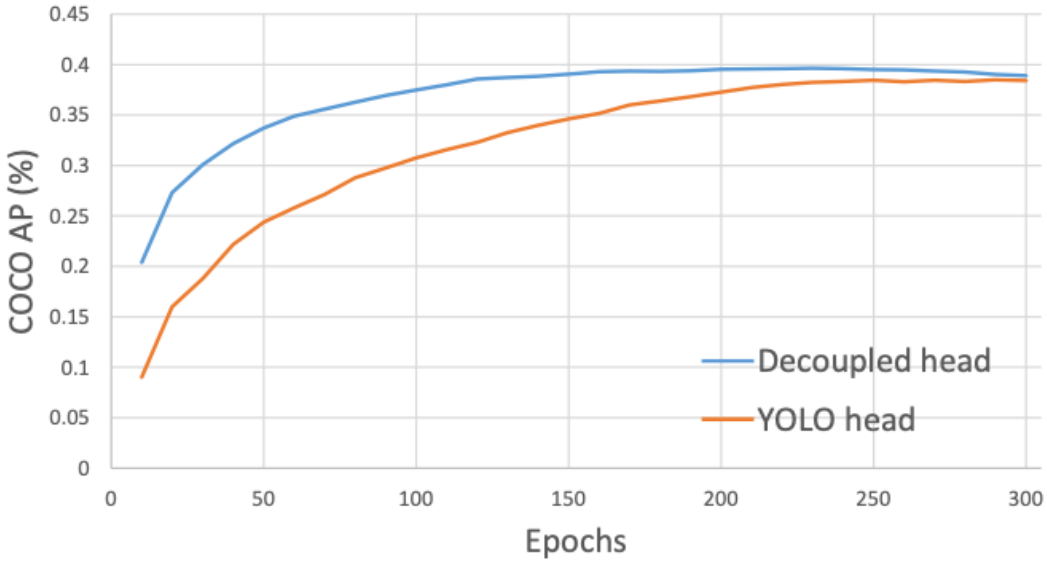

Figure 4.7: YOLOX authors experiment with training a YOLOv3 object detector with coupled and decoupled heads. The object detector converges towards optimum significantly faster with the decoupled head and also achieves better accuracy [11].

The third and the last significant improvement is the SimOTA label assignment strategy [11]. SimOTA is based on the optimal transport assignment for object detection (OTA) research from the same authors as the YOLOX object detector [10]. SimOTA is used to optimize YOLOX training process by calculating unit transportation cost between the predictions and ground truth labels as the weighted sum of the classification and regression losses. This makes the training process faster and also increases the detection performance.

4.3 YOLOX Performance

With the latest updates to the YOLO architecture, YOLOX offers the best speed and accuracy that the YOLO models have reached by August 2021 [11].

Figure 4.8 presents a speed and precision comparison between several real-time object detectors. When comparing similar size YOLOv5 and YOLOX models, the latter offers 0.8 % to 2.0 % higher average precision. Even though the difference between these two models is not as large as one might expect, it is good to remember how these numbers are reached.

As discussed in chapter 4.2, YOLOX is the first anchor-free YOLO object detector. As the YOLOX is anchor-free architecture, it requires only a minimal amount of hyperparameter tuning to reach the optimal performance. As the previous model's used anchor-based localization, the user had to pre-define the anchors before the actual model training. At the same time, pre-defining optimal anchors is not a straightforward or simple task. For example, YOLOv3 authors use unsupervised clustering methods to find the optimal anchor parameters [34].

Method	Backbone	Parameters	Resolution	FPS (V100)	AP (%)	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
EfficientDet-D0	Efficient-B0	-	512x512	98.0	33.8	52.2	35.8	12.0	38.3	51.2
EfficientDet-D1	Efficient-B1	-	640x640	74.1	39.6	58.6	42.3	17.9	44.3	56.0
EfficientDet-D2	Efficient-B2	-	768x768	56.5	43.0	62.3	46.2	22.5	47.0	58.4
EfficientDet-D3	Efficient-B3	-	896x896	34.5	45.8	65.0	49.3	26.6	49.4	59.8
YOLOv3+ASFF	Darknet-53	-	608x608	45.5	42.4	63.0	47.4	25.5	45.7	52.3
YOLOv3+ASFF	Darknet-53	-	800x800	29.4	43.9	64.1	49.2	27.0	46.6	53.4
YOLOv3-ultralytics	Darknet-53	-	640x640	95.2	44.3	64.6	-	-	-	-
PP-YOLOv2	ResNet50-vd-dcn	-	640x640	68.9	49.5	68.2	54.4	30.7	52.9	61.2
PP-YOLOv2	ResNet101-vd-dcn	-	640x640	50.3	50.3	69.0	55.3	31.6	53.9	62.4
YOLOv4	CSPDarknet-53	-	608x608	62.0	43.5	65.7	47.3	26.7	46.7	53.3
YOLOv4-CSP	Modified CSP v5	-	640x640	73.0	47.5	66.2	51.7	28.2	51.2	59.8
YOLOv5-S	Modified CSP v5	7.3 M	640x640	114.9	36.6	-	-	-	-	-
YOLOv5-M	Modified CSP v5	21.4 M	640x640	90.1	44.5	63.1	-	-	-	-
YOLOv5-L	Modified CSP v5	47.1 M	640x640	73.0	48.2	66.9	-	-	-	-
YOLOv5-X	Modified CSP v5	87.8 M	640x640	62.5	50.4	68.8	-	-	-	-
YOLOX-DarkNet53	Darknet-53	-	640x640	90.1	47.4	67.3	52.1	27.5	51.5	60.9
YOLOX-S	Modified CSP v5	9.0 M	640x640	102.0	36.9	-	-	-	-	-
YOLOX-M	Modified CSP v5	25.3 M	640x640	81.3	46.4	65.4	50.6	26.3	51.0	59.9
YOLOX-L	Modified CSP v5	54.2 M	640x640	69.0	50.0	68.5	54.5	29.8	54.5	64.4
YOLOX-X	Modified CSP v5	99.1 M	640x640	57.8	51.2	69.6	55.7	31.2	56.1	66.1

Figure 4.8: Speed and accuracy comparison on COCO 2017 test-dev dataset [11].

Interestingly, YOLOX was originally designed for the CVPR 2021 streaming perception challenge [45]. In this challenge, algorithm latency was scored together with the model accuracy. The YOLOX team received 1st place in this competition by using the YOLOX-L model with a high-resolution input image size [11].

5. Active Learning

We need a large labeled dataset to train supervised machine learning. Having a large dataset may not be enough, as the dataset also needs to cover the whole variance of the data population. In other words, your labeled dataset needs to have quality and quantity [14].

In the computer vision context, quality means that images need to cover a wide variance of how the objects look. For example, if we want to train cat vs. dog classifiers, we need to have labeled training images from a wide range of dog and cat breeds. Also, cats and dogs should be in pictures in a wide range of positions, and photos should have been taken from a wide range of different angles.

Annotating images is a time-consuming task [6]. The high price of the annotation process is even more apparent in cases where we use a team of highly trained professionals to ensure the quality of the annotated dataset. While the annotation process is time-consuming and expensive, every individual image does not offer the same amount of new information [14]. One such example is when the dataset has multiple identical or almost identical images, like in the figure 5.1. In this case, it is enough to annotate and train the model with only one of the images belonging to this group. Annotating all the identical or almost identical images would not increase the model's performance, and it would be an inefficient use of the annotation resources.

Active learning is a supervised learning technique that tackles the inefficiency of collecting and annotating irrelevant data samples [20]. In the active learning, we actively choose the images we will annotate and add to the training dataset [30]. The aim is only to pick the most informative data samples and ignore the ones that would not improve the model performance. As a result, the amount of data labeling work can be reduced [14].

Even though active learning has been proved to be an effective method to train an image classification model, there is only a minimal amount of research on applying active learning to the object detection [20][14]. Also, according to the questionnaire by Tomanek and Olsson 2009 proved that active learning is not widely used [30]. We can argue that active learning has not received the attention it deserves as an efficient data annotation process is an essential but time-consuming task.

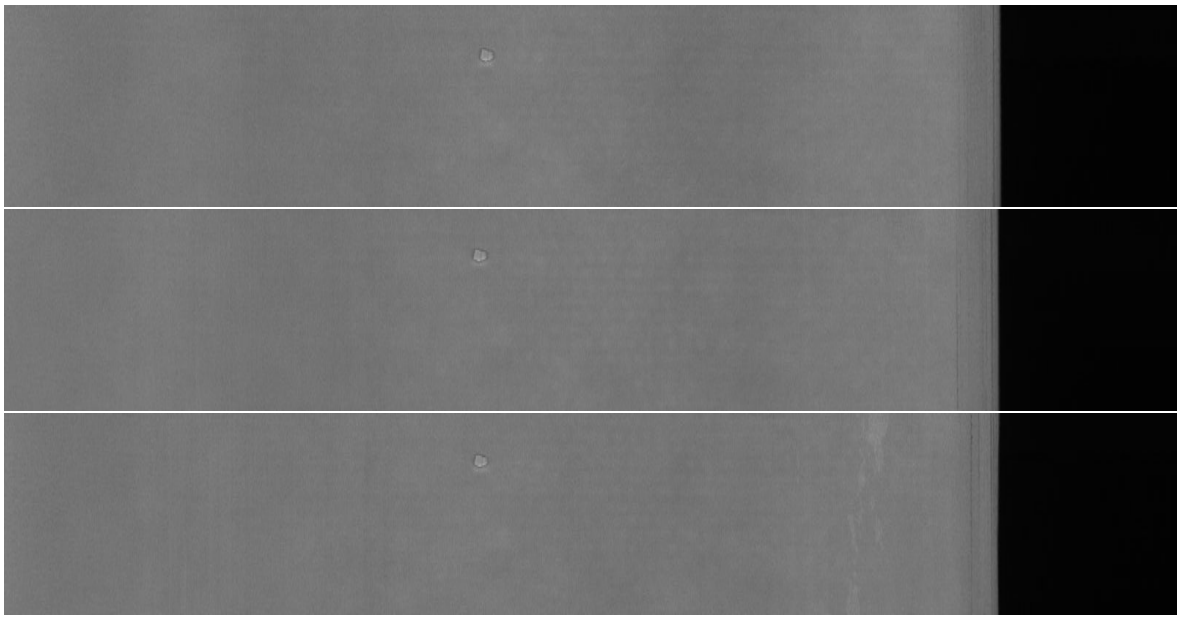

Figure 5.1: Three almost identical images from the surface of the stainless steel. The light round area is a surface defect called indentation.

5.1 Active Learning vs. Passive Learning

The typical way of annotating data is called passive learning. In passive learning, we take a random subset of the data to be labeled, or if the dataset is small enough, then label the whole data set. In other words, we do not actively select images that we will annotate.

With passive learning, we may face several problems. The first one is taking samples that do not benefit the model accuracy or even decrease the model performance [42]. As introduced in the previous chapter, this could mean picking several similar examples that do not have any new information.

The second possible problem is data balance. When we want to train a machine learning model that does classification, we want to have balanced training dataset. Unfortunately, the real-life unlabelled dataset is commonly not balanced [42]. It is common that we can get samples of samples more easily than other and we may even have a situation where most of the samples belong to a single class and there are only few samples from the other classes.

For example, let's consider a situation where we want to build an animal classifier. We take all the animal images we can find from the internet and create a large pool of unlabelled animal images. Let's say that the unlabelled data pool has images from 100 different animal species, including dogs, cats, ice bears, mooses, foxes, and others. The number of images per class is naturally unbalanced, as it is easier to take images from cats than from the ice bears.

Let's say that the unlabelled data pool D_U is so large that we are not able to label all the images, but instead, we can only label a random sample of the images in the unlabelled data pool. Let's consider that we have a prior estimate, that if we randomly select image from the unlabelled datapool, then $P(dog) = \frac{1}{3}$, $P(cat) = \frac{1}{3}$, and $P(Others) = \frac{1}{3}$. In the class "Others," we have all the other 98 animal species. If we labeled the data with passive learning, it would be probable that we would mostly pick dog and cat images, and the trained classifier would perform poorly in classifying other animal species. Also, we would probably label many images that do not provide any new information. We can solve this problem with active learning by picking only the most informative samples and picking images from all the classes [6].

The primary purpose of active learning is to offer an effective approach for data annotation [20]. As a result, the performance of active learning is often measured by how much data is required to reach some given performance level when data is sampled from the same set of unlabeled data by passive learning or by active learning [30]. In our research in chapter 6, we will show how with a maximum uncertainty active learning, we can reduce the number of labeled images from 16,551 to 13,473 images, while model mAP_{50} drops only from 83.6 % to 83.2 %.

5.2 Active Learning Process

Active learning is a class of algorithms that iteratively search for the most informative samples to be added to the training data set [20]. As it is a class of algorithms, there is more than one way to apply active learning, but next, we will introduce the standard active learning loop.

The standard active learning loop goes as follows:

1. Collect unlabelled dataset D_{Ui}.
2. From D_{Ui}, select and label samples S_i.
3. Remove labelled samples S_i from the unlabelled dataset D_{Ui}. Now, labelled samples S_i will create the first labelled dataset D_{Li}. The first labeled dataset D_{L0} is called the seed set.
4. Train a model M_i with the labelled dataset D_{Li}.
5. Apply model M_i to the unlabelled dataset D_{Ui}.
6. For each unlabelled sample in the dataset D_{Ui}, estimate whether this sample contains information that model M_i has not been learned before.

7. Select the most informative samples S_{i+1} for the Oracle for annotation. Oracle is someone who can give the ground truth answers for unlabelled data. Usually, Oracle is a human annotator.

8. Add labelled samples S_{i+1} to the labelled dataset D_{Li}. Also, remove samples S_{i+1} from the unlabelled dataset D_{Ui}.

9. Repeat steps 4 through 8, until unlabelled dataset D_{Ui} is empty, or until some stopping criterion is met. These steps can be found as a image from the figure 5.2.

10. Output a model that is trained with the labelled data D_{Lj}, where j is the number of active learning loops. Through the whole process, $D_{Ui} \cap D_{Li} = \emptyset$ and size of the dataset D_{Li} increases and size of the dataset D_{Li} decreases (if new unlabelled data is not collected during the process).

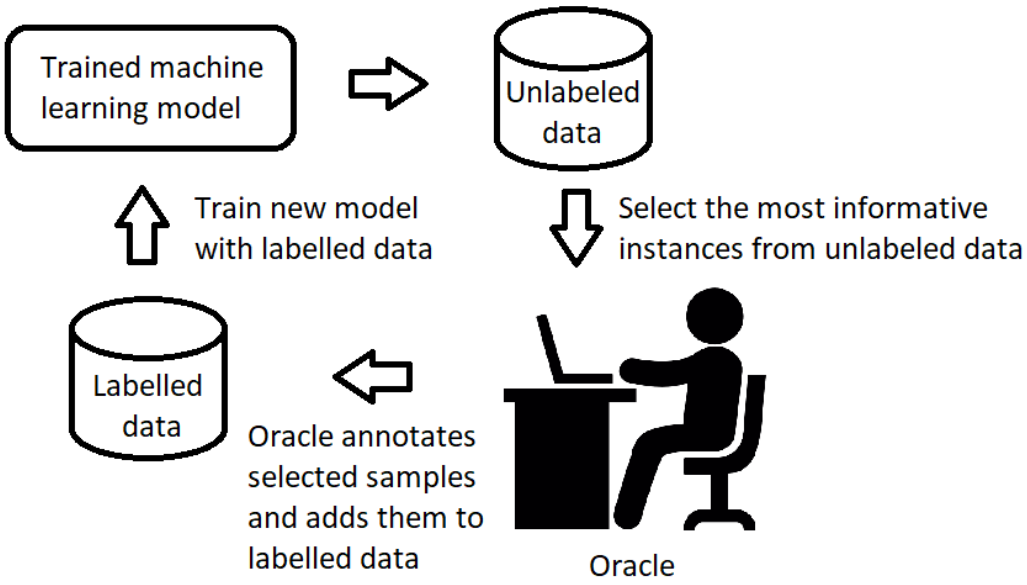

Figure 5.2: The common active learning loop as a image.

The active learning loop is relatively standard on the theoretical level, but there may be lots of variance at the practical level. In real-life applications, it is expected that the unlabelled dataset is not fixed. Instead, it may be continuously growing and so large that it would not be feasible to analyze each data sample in each active learning loop round.

Next, we will introduce the three most important steps of the active learning loop on a more detailed level. These are: defining the seed set, finding the most informative data samples, and terminating the active learning loop.

5.2.1 Defining the seed set

As we can see from the previously introduced list, the first part of the active learning loop is to create labeled seed data set [20]. The simplest way to generate seed data set is to take and label a random subset from the unlabeled data. The base data set should have multiple samples from each class in the optimal situation. If class sizes in the unlabeled data set are heavily unbalanced, getting samples from each class may not be easy. In this case, we may use clustering, the unsupervised learning technique, to cluster images and then pick samples from each cluster [30].

The benefits of creating seed set by clustering have not been studied in computer vision. In the context of natural language processing, its benefits have not been noticeable when compared to random sampling [29]. We would still expect deep learning-based image clustering to be a powerful tool for creating the seed set, but verifying this question is outside of our own experiment.

5.2.2 Selecting the most informative data samples

Finding the most informative data samples in the active learning loop can be manual or automatic. First, we will introduce the manual process. Then we will introduce the most common automatic ways to pick the most informative data samples.

In manual active learning, the data labeler manually inspects the model results. In other words, the data labeler manually looks through the model outputs and picks the data samples where the model has made a mistake. When the data labeler finds a data sample where the model has made a mistake, he knows that this data sample has new information that should be labeled and added to the training dataset.

Next, we will introduce the three most common active learning approaches [30]. These are:

1. Query by uncertainty
2. Query by committee
3. Active learning with redundant views

The first approach is called query by uncertainty, also known as uncertainty sampling or uncertainty reduction [30]. Query by uncertainty is the most straightforward active learning method, as it uses the model prediction confidences as its main source to find the most informative samples. When the model makes a high confidence prediction, it indicates that the given sample has similar information as the data used to train the model. Conversely, when the model makes a low confidence prediction, the

given instance has new and valuable information and should be annotated and added to the training dataset. Query by uncertainty can be applied to a wide range of machine learning and deep learning models, but the model needs to provide confidence scoring to indicate how confident it is in each prediction it performs.

The second approach is called query by committee. In this context, committee members are different predictive models. In the query by committee, committee members make predictions on the same data. When predictions between committee members differ, then the given sample includes new and valuable information [30]. The use of query by committee expects that the used models are not identical, as identical processes would output identical predictions, and there would be no disagreements among the committee members.

The third and the last approach of active learning techniques is called active learning with redundant views. Redundant views are similar to the query by committee, as both techniques aim to find data samples that will cause disagreement among the committee members [30]. While in the query by committee, the committee members use the same input data, but in the redundant views, committee members use different input data that should give the same output. In other words, the domain has redundant views if the domain has at least two mutually exclusive sets of features that can be used to learn the same target concept [27]. Then we can train two or more models that should give the same output by using different input data. For example, we can use images and lidar data to estimate object dimensions in 3D.

These are the main three approaches to active learning. In our research in chapter 6 we study uncertainty sampling in object detection. The reason is that it is computationally the most efficient active learning method, and our experiment domain does not have two or more mutually exclusive sets required for the redundant views.

In our proposed method, we define detected object uncertainty as a confidence score. A low confidence score indicates that the image has new, not yet learned information. We define the total object confidence score for a single detected object (det_i) as follow:

$$S_{total}(det_i) = \max_{c_{cls} \in K} \hat{p}(c_{cls}|det_i) * IoU(det_i), \tag{5.1}$$

where K is the set of possible object classes, $\max_{c_{cls} \in K} \hat{p}(c_{cls}|det_i)$ is the classification confidence of the most confident object class for detection det_i, and $IoU(det_i)$ is the predicted IoU of detection det_i. This follows the principles used in the FCOS [40] and YOLOX [11] to filter out low confidence detections, but we use it to find the most informative unlabelled images. To our knowledge, we are the first ones to use this technique in the active learning context.

5.2.3 Terminating the active learning loop

So far, we have introduced the first two main parts of the typical active learning process: creating the seed set and selecting the most informative samples. Then the last step is to decide when to terminate the active learning process. As one main purpose of active learning is to minimize the amount of data labeling work, we need to know when to stop the loop. From the theory perspective, the active learning loop should be terminated when the model reaches its peak effectiveness. However, it is challenging to know when this happens before all the data is labeled [48].

Determining when to stop the active learning process has no unequivocal answer, and according to Zhu et al., it is not a very well-studied topic [48]. Terminating the active learning process is also out of the scope of our study, but we will introduce the main terminating approaches. We did not find any research of the terminating approaches in the deep learning context, but from the theory perspective, they should also be applicable to both machine learning and deep learning models.

The different terminating approaches can be grouped under the three main categories. These are:

1. Resources defined terminating approach
2. Model performance defined terminating approach
3. Statistical terminating approach

In the resources defined terminating approach, the annotation resources define when to end the active learning process. In practice, this could mean that we have annotation resources to perform an active learning loop ten times, annotate 1,000 data samples each time, and therefore label a total of 10,000 data samples. However, there is no way to estimate the optimum training data size beforehand, so we might stop the process too early or too late [48]. Even the active learning with resources defined terminating strategy may not end the optimum model performance, it is still a significantly better approach than passive learning, where we randomly pick the data we annotate [30].

Then the second approach is to use model performance-defined stopping criterion. In this approach, we evaluate the model performance in each iteration and repeat the active learning loop until we reach the predefined model performance or we are satisfied with the model performance. To effectively use this approach, we need to evaluate model performance reliably in each iteration, which is not a trivial task, especially if the test dataset is relatively small. Secondly, it is difficult to predefine an appropriate and achievable performance because it depends on the problem and users' requirements [48].

The last category of the stopping criterion is called the statistical terminating approach. In the statistical terminating approach, we aim to estimate the optimum active learning process stopping moment by using the unlabelled data or by using the trend of the model performance development. As in the active learning process, the model outputs the estimated informativity for each data sample. We can use this information to estimate when no informative samples are left [48]. We can also use performance development through an active learning loop to estimate when the model reaches the peak performance. The theory behind the model performance development approach goes as follows. When the size of the labeled training data increases, then the model performance also increases. While the model performance increases, it also converges towards the model maximum performance level. This means that the model performance improvement decreases when the number of training images increases. We can see an example in figure 6.8, where we used a predefined and fixed performance test set. In this example, the performance converges towards a value of 0.84. Using the model performance development as a stopping criterion has not been formally studied, but from the theory perspective, we can agree with Zhu et al. [48] that it is probably an efficient stopping criterion.

We have now introduced the main strategies on how to define the active learning stopping criterion. The categories are loosely defined, and the categories are formed by the similarity of the individual techniques. We believe that the optimum stopping criterion would be a combination of multiple individual techniques. Then we can get the best benefits out of the whole active learning process.

As we discussed at the beginning of the chapter, terminating the active learning process is not a very well-studied topic, and we did not find any research that would have been done in deep learning or in the field of computer vision. Nevertheless, we believe that they should be applicable also to modern neural networks.

5.3 Active learning as annotation support

The main benefit of active learning is to choose the most informative samples from the unlabelled data pool, but selecting the most informative samples is not the only potential benefit in the active learning loop. We can improve the typical active learning loop by pre-annotating the unlabelled data samples before giving them to Oracle. Then, Oracle only needs to confirm that labeling is correct and fix the potentially incorrect samples in this situation.

Object detection has one of the most time-consuming data annotation processes in the context of computer vision. In this process, the data labeler first needs to search all the interesting objects in the image, draw tight bounding boxes around each object

Figure 5.3: Image without pre-annotation.

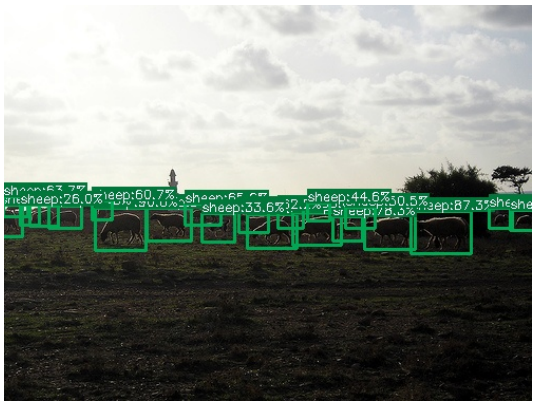

Figure 5.4: Image with pre-annotation.

and classify objects inside the bounding boxes. This process needs to be done carefully, as images may have objects that are important but difficult to notice. According to Su et al., annotating a single object from an image takes on average 50.8 seconds, but verifying labeling quality takes on average only 21.9 seconds [39].

In figures 5.3 and 5.4, we present examples of how pre-annotation helps to detect all the relevant objects. The image has multiple sheep, and the objects are relatively small. Even though the pre-annotation may not be completely correct, it highlights most of the relevant objects, and therefore annotator may focus on fixing the annotation instead of searching for the sheep from the image.

In figure 5.5 we present an example of using uncertainty sampling and image pre-annotation. Image has been automatically selected for the annotator because the object detector detected an object with low confidence. In this case, the annotator only needs to remove the incorrect tv monitor labeling and slightly adjust the cat bounding box to make it tighter.

In summary, by having data pre-annotation as an active learning process, the role of the image labeling expert changes. Without the pre-annotation, the annotator focuses on finding objects, drawing bounding boxes, and classifying objects. But when using pre-annotation made by the object detector, the annotator only needs to verify that annotations are correct and fix the incorrect labels. The second option takes less time and helps the annotator find all the relevant objects, improving the annotation quality.

Figure 5.5: Example of the image pre-annotation in the active learning loop. Image has been automatically selected because the object detector made a low confidence detection. Then, model detections are used as the pre-annotation. In this example, cat pre-annotation is correct, but the kitchen cabinet labeled as a tvmonitor needs to be removed.

6. Experiments

We propose an uncertainty sampling-based active learning method for object detection. As we want to keep our method applicable for real-life use cases, our proposed active learning method is designed to require only a minimal computation overhead and keep the labeled training dataset balanced.

Active learning aims to minimize the number of labeled training samples, and this task can be split into three subparts. These are: defining the seed set, sampling method, and the stopping criterion. Our proposed method is only meant to answer the question of the optimal sampling method, and we do not experiment with these two other questions.

We use the YOLOX-S (YOLOX small) object detector in our experiment, but the method is also applicable to other object detectors that predict localization IoU for every detection. Our method is unsuitable for the anchor-based object detectors, as these do not usually output localization confidence, which is required to calculate detection scores.

The experiment chapter follows the following structure. First, we introduce the used dataset and the YOLOX-S object detector. Then we introduce the active learning process step-by-step, and finally, we introduce the results.

6.1 Pascal VOC object detection dataset

We use the Pascal VOC object detection dataset [8]. We want to make our experiment setup as realistic as possible, and that's why we want to use a larger dataset than commonly used VOC 2007 or VOC 2012. As a solution, we combine the VOC 2007 train+validation+test dataset with the VOC 2012 train+validation dataset. As a result, we get a dataset with 21,503 images and 33,264 objects.

Then we split this dataset into two parts. This first part takes all the training and validation images from VOC 2007 and VOC 2012 datasets. This dataset creates the unlabelled dataset D_{Ui} where we aim to sample the most informative samples. The second part takes all the images from the VOC 2007 test dataset. We call this dataset a performance test set or ground truth test set. It is used to reliably evaluate our

model performance changes through the 10 rounds of the active learning loop. All the details of the used dataset are presented in figure 6.1.

How used	As train, validation and test	As performance test
Data source	VOC2007+2012 Train+Val	VOC2007 Test
Total images	16551	4952
aeroplane	916	205
bicycle	826	250
bird	1106	289
boat	704	176
bottle	1030	240
bus	627	183
car	1990	775
cat	1428	332
chair	1870	545
cow	455	127
diningtable	904	247
dog	1727	433
horse	777	279
motorbike	785	233
person	6469	2097
pottedplant	841	254
sheep	423	98
sofa	1067	355
train	813	259
tvmonitor	874	255
Total objects	25632	7632

Figure 6.1: Number of images and objects in unknown data set and performance test data set.

When we train the object detector with the labeled dataset, we use the following data split: 70 % to the train set, 15 % to the validation set, and 15 % to the test set. The performance test set is kept separately, as we want to have a fixed test set to reliably evaluate the model performance through the whole active learning process.

6.2 YOLOX-S object detector

YOLOX-S is the small-size version of the YOLOX object detector [11]. The difference between the different size YOLOX versions can be seen from figure 4.8 where we compare a number of parameters in different YOLOX models.

As discussed in chapter 4.1, a different number of parameters is achieved by scaling the Modified CSP v5 backbone. In practice, this means the number of convolutional kernels and layers used in the backbone.

We use the YOLOX-S in our experiment, as it is faster to train than the larger YOLOX models. Nevertheless, the single model training takes approximately 6 to 12 hours, depending on the amount of the training data, when the model is trained with a cluster of 4 pieces of Tesla M60 GPUs.

6.3 Model training

When we start to train the object detector, we use transfer learning to transfer knowledge from one domain to our task and that way to improve the trained model [28]. We first tested the transfer learning to transfer knowledge of the YOLOX-S object detector trained with the COCO object detection dataset to train a object detector for Pascal VOC object dataset. However, we found that these two datasets contain mostly the same classes, and the actual learning from to Pascal VOC dataset was minimal. Because we aimed to experiment the uncertainty sampling in a real-life situation, where it is likely that the gap between the task domains is large, we decided not to use transfer learning from the model trained with the COCO dataset. As a solution, we decided to use transfer learning from the YOLOX-S model trained with the steel dataset. This is a private dataset of the Outokumpu Oyj, and the dataset contains images of the steel surface defects. Now the domains were different enough for our experiment purposes. In figure 6.2, we compare the results when the model is trained with 2,000 Pascal VOC images, and one model uses transfer learning from COCO and another from the Steel dataset.

In the model training we follow the practices proposed by the YOLOX developers [11]. The model is trained with 300 epochs, with batch size of 32 images. We used all augmentations as introduced in the chapter 2.2, but we removed mixup and Mosaic augmentations from the last 15 epochs. This follows the recommended training process.

We evaluate the model performance with the validation dataset in every 10 epochs in the training process. We save these evaluated models, and the best performing model is selected as the output for the given training process.

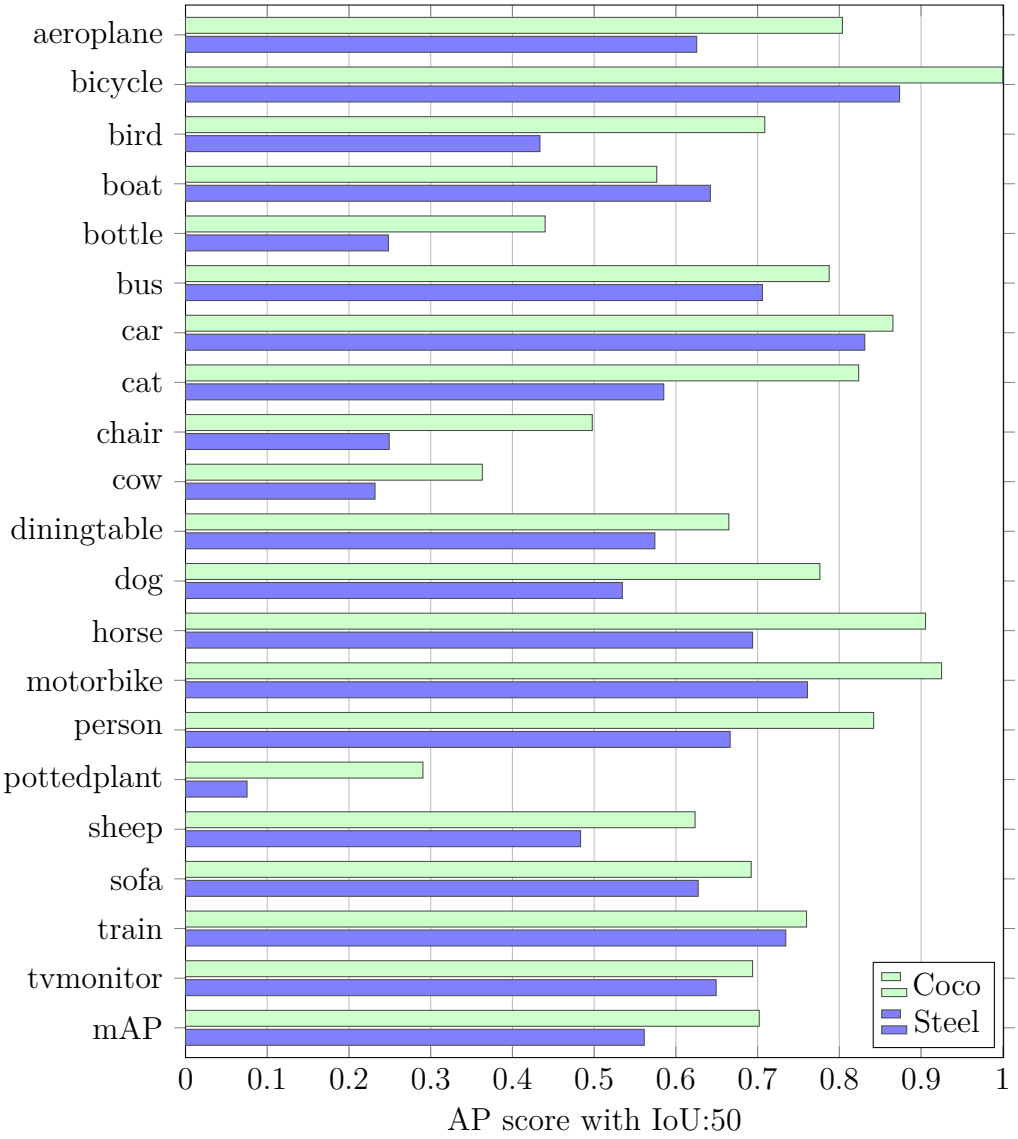

Figure 6.2: Test set results with two pretrained models re-trainned with same dataset. Models are trained with 2,000 Pascal VOC images.

6.4 Defining detection uncertainty score

YOLOX is an anchor-free object detector, and therefore it natively outputs two confidence scores. As can be seen from figure 4.6, these are the classification confidence, and the second is IoU confidence. As described in chapter 5.2.2, the predicted IoU can be used to estimate the localization confidence score. YOLOX provides these two confidence numbers for every detection, so using them does not cause any additional computing overhead.

Because we want to measure and compare the detection confidences, we define the score for each detection. The combined confidence score formula is introduced in detail in chapter 5, but the high-level idea is following:

$$S_{total}(det_i) = S_{cls}(det_i) * IoU(det_i), \tag{6.1}$$

where det_i is a single detection, $S_{cls}(det_i)$ is the classification confidence for detection det_i, and $IoU(det_i)$ is the predicted bounding box IoU for detection det_i.

We attach the score calculation as a part of the YOLOX-S inference process. In practice, we save detection results to the XML file for each image, and the XML follows the standard Pascal VOC format. An example of the output is presented in figure 6.3. Then after we have processed all the images, we can easily create a summary of all the detection and select the most informative samples.

6.5 Used Active Learning Process

In the experiment, we follow the standard active learning loop process. Then to select the most informative samples, we use detection score as defined in chapters 5.2.2 and 6.4.

Step-by-step, the used active learning process went as follow:

1. Collect unlabelled dataset D_{Ui}. D_{Ui} is combination of VOC2007+2012 train and validation sets. Details are presented in the figure 6.1.

2. From D_{Ui}, select and label samples S_i. We select randomly 2,000 images for a seed set. Seed set details can be found from the figure 6.4.

3. Remove labelled samples S_i from the unlabelled dataset D_{Ui}. Now, labelled samples S_i will create the first labelled dataset D_{L0}.

4. Train a model M_i with the labelled dataset D_{Li}.

5. Apply model M_i to the unlabelled dataset D_{Ui}, calculate detection scores, and save detections with score greater than threshold $t > 0.2$. We use a lower threshold t to increase the probability of sampling objects from wanted object classes. This process is also presented as a pseudo-code in pseudo-code 1.

6. Select 100 lowest score detections from the each object class as described in the pseudo-code 2.

7. Annotate selected images (in experiment we use original Pascal VOC dataset labels).

8. Add labelled samples S_{i+1} to the labelled dataset D_{Li}. Also, remove samples S_{i+1} from the unlabelled dataset D_{Ui}.

9. Repeat steps 4 through 8, until unlabelled dataset D_{Ui} is empty.

Algorithm 1 Pseudo-code for running object detection and saving results.

Input: Unlabelled image pool [D_U], Trained object detector [Det]
Output: Saved object detections [$SavedDetections$]

for *image i in* D_U **do**
  $Detections_i \leftarrow Det(i)$
  for $Detection_i$ *in* $Detections_i$ **do**
    $UncertaintyScore_i \leftarrow LocalizationConfidence_i * ClassificationConfidence_i$
    if $UncertaintyScore_i > 0.2$ **then**
      $SavedDetections \leftarrow Save(Detection_i)$
    end if
  end for
end for
Return: $SavedDetections$

Algorithm 2 Pseudo-code for select the most informative images.

Input: Saved object detections [$SavedDetections$]
Output: Selected images names list [$ImageNamesList$]
for $ObjectClass\ c\ in\ SavedDetections$ **do**
  $Detections_c \leftarrow filterWithClass(SavedDetections)$
  $Threshold_c \leftarrow getUpperThreshold(Detections_c)$ {100th lowest confidence}
  for $Detection_i\ in\ Detections_c$ **do**
    if $DetectionConfidence_i < Threshold_c$ **then**
      $ImageNamesList \leftarrow ImageNameToList(Detection_i)$
    end if
  end for
end for
Return: $ImageNamesList$

6.6 Training rounds progress

Our experiment consists of 10 training rounds, as described in section 6.5. The only difference is the last training round, where we do not apply any confidence filtering but instead select all the unlabelled images that are left.

In each training round, we split our labeled data into three parts: training, validation, and test sets. We use a stratified split method to keep object classes in each set balanced. We use a 70-15-15 split, where 70 % go to the training set, 15 % to the validation set, and 15 % for the test set on each training round. Details of the number of images in each set in each training round are described in figure 6.4.

In figure 6.5, we can see the used detection upper confidence filtering thresholds for each object class on each training round. As described before, the score is defined as the 100th lowest detection score. Then we select all the images with detection scores below the defined upper threshold.

As we can expect, the upper confidence threshold to select 100 lowest confidence detections from each object class rises through the training rounds. Details can be found in figure 6.5. This is due to two reasons. First, we select the most difficult images on each training round and, therefore, the number of difficult unlabelled images gets smaller and smaller. Secondly, the model gets better on each training round, which reduces the number of low confidence detections.

In figure 6.6, we can see how the class-specific AP50 score rises quickly during the first training rounds. Quick model improvement is a behavior we want to see in the effective active learning process, as our aim is to get the best possible results with the least amount of data.

Our method is designed to keep the labeled training dataset as balanced as possible. To achieve this task, we select 100 low score object detections from each class on each training round. Because of this number of objects from each object class grows approximately by 100 pieces. We want to highlight the word approximately because of the two reasons. First, we sample images with the low confidence detection, and therefore these detections, by default, have errors. In figure 5.4, we can see an example where the model has falsely detected a tv monitor from the image. Secondly, Pascal VOC dataset images may have multiple objects and multiple object classes in a single image. Therefore several low confidence detections can be in the same image. Also, high confidence detection may appear in the same image with a low confidence detection, as in the figure 5.4. In figure 6.7, we can see the number of labeled objects from each class in each training round. Our seed set was a random sample of 2,000 images from the unlabelled images, so at the beginning, the number of objects in different object classes differs.

When we combine the information from figures 6.6 and 6.7, we can see that some object classes are easier to detect than others. For example, in the case of aeroplane the AP50 score reaches 80 % with 278 labeled objects, but with bird, we need 787 labeled objects to achieve a similar AP50 score. The probable reason is the intra-class variance, the variance of what objects can look like, but the deeper investigation of the reason is outside of the scope of our research.

The working active learning solution maximizes the model performance with minimal annotated data samples. From figure 6.8, we can see that the model performance increases quickly during the first training rounds, but then the slope tapers. This is expected behavior, as the images in the unlabelled data pool get less informative after each round and model performance convergence towards the optimal maxima. Based on the AP50 score of the test set and performance test set, the progress is barely noticeable after the training round 8th.

6.7 Results

With 16,551 images, the MAP50 score is 0.8361. With 13,473 images result is almost the same, 0.8315. So we could remove 3078 images, barely noticing any difference in results. By removing 5559 images, the performance drop is still only 2.21 %. Object detection annotation is a relatively slow process, so if we expect that annotating a single image takes 1 minute, then by not annotating the removed 5559 images, we would have saved close to 100 hours. Also, this method is computationally extremely efficient and easy to use.

This is the first study where the predicted IoU is used as a metric to measure localization confidence. We have proved that when predicted IoU is combined with the classification confidence, we can efficiently sample the most informative images from the pool of unlabelled data.

As we can see from figure 6.8, the greatest benefits from our active learning method can be achieved in the early training rounds. This proves that the method efficiently samples the most informative images because otherwise, the improvements would not be as drastic.

According to Haussmann et al., the images in the Pascal VOC dataset are pre-selected when the dataset was created and thus contain mostly informative images [14]. Based on this statement, we could expect even better results with our method when applied to the real-life dataset.

```
<annotation>
  <filename>2007_000017.jpg</filename>
  <analysis_time>2021-11-27T13:41:49.897255</analysis_time>
  <size>
    <width>480</width>
    <height>364</height>
    <depth>3</depth>
  </size>
  <object>
    <name>horse</name>
    <confidence>85.2</confidence>
    <bndbox>
      <xmin>123</xmin>
      <ymin>83</ymin>
      <xmax>429</xmax>
      <ymax>351</ymax>
    </bndbox>
  </object>
  <object>
    <name>person</name>
    <confidence>64.9</confidence>
    <bndbox>
      <xmin>166</xmin>
      <ymin>63</ymin>
      <xmax>286</xmax>
      <ymax>204</ymax>
    </bndbox>
  </object>
</annotation>
```

Figure 6.3: YOLOX inference output XML.

	Training round									
Dataset name	1	2	3	4	5	6	7	8	9	10
Train+Val+Test	2000	3816	5638	7453	9214	10922	12322	13473	14217	16551
Unknown	14551	12735	10913	9098	7337	5629	4229	3078	2334	0
Train	1400	2671	3946	5217	6449	7645	8625	9431	9951	11585
Validation	300	572	846	1118	1382	1638	1848	2021	2133	2483
Test	300	573	846	1118	1383	1639	1849	2021	2133	2483
Performance test	4935	4935	4935	4935	4935	4935	4935	4935	4935	4935

Figure 6.4: Number of images in training rounds.

	Confidence level filter upper threshold								
Object	1 to 2	2 to 3	3 to 4	4 to 5	5 to 6	6 to 7	7 to 8	8 to 9	9 to 10
aeroplane	37.0%	32.0%	60.7%	74.2%	84.6%	87.4%	89.4%	91.7%	N/A
bicycle	44.0%	56.0%	80.1%	86.4%	91.4%	93.6%	96.2%	100%	N/A
bird	34.9%	26.1%	31.8%	43.3%	57.8%	76.6%	83.7%	88.3%	N/A
boat	29.5%	26.3%	33.4%	44.6%	75.1%	88.8%	100%	100%	N/A
bottle	28.5%	27.7%	30.1%	34.8%	55.7%	84.2%	92.7%	100%	N/A
bus	56.7%	73.9%	85.4%	92.1%	95.3%	100%	100%	100%	N/A
car	24.9%	24.0%	25.3%	29.3%	40.7%	67.5%	82.0%	91.2%	N/A
cat	34.7%	37.7%	50.7%	65.1%	75.3%	78.2%	82.7%	86.3%	N/A
chair	23.5%	23.1%	25.5%	28.7%	33.2%	64.4%	85.5%	100%	N/A
cow	49.3%	49.6%	67.3%	85.7%	100%	100%	100%	100%	N/A
diningtable	44.6%	40.8%	58.4%	83.2%	100%	100%	100%	100%	N/A
dog	31.8%	31.5%	44.1%	55.6%	65.3%	76.3%	82.3%	86.6%	N/A
horse	52.3%	63.1%	73.8%	84.9%	89.5%	92.5%	94.8%	100%	N/A
motorbike	59.8%	46.5%	76.8%	86.8%	89.4%	92.2%	95.5%	100%	N/A
person	21.2%	21.0%	21.4%	22.1%	23.6%	25.5%	30.1%	45.1%	N/A
pottedplant	28.5%	27.2%	29.2%	39.8%	58.5%	84.2%	100%	100%	N/A
sheep	34.7%	50.2%	66.0%	88.7%	100%	100%	100%	100%	N/A
sofa	31.8%	43.5%	50.6%	73.3%	87.4%	92.9%	100%	100%	N/A
train	40.7%	37.4%	55.8%	78.1%	85.9%	89.1%	92.5%	100%	N/A
tvmonitor	30.5%	35.9%	41.5%	54.8%	81.9%	90.5%	100%	100%	N/A
Average	36.9%	38.7%	50.4%	62.6%	74.5%	84.2%	90.4%	94.5%	N/A
Median	34.7%	36.7%	50.7%	69.2%	83.25%	88.9%	95.15%	100%	N/A

Figure 6.5: Confidence filter upper threshold.

	AP50 score for each class after each training round									
Object	1	2	3	4	5	6	7	8	9	10
aeroplane	0.6140	0.7334	0.8208	0.8400	0.8681	**0.9015**	0.8869	0.8985	0.8824	**0.9079**
bicycle	0.6992	0.7793	0.8094	0.8720	0.8854	0.9031	0.9068	**0.9293**	0.9113	**0.9216**
bird	0.3706	0.5193	0.6193	0.7197	0.7302	0.7781	0.7911	0.8052	**0.8245**	**0.8391**
boat	0.4929	0.5672	0.6435	0.6861	0.6765	0.7311	0.7385	**0.7605**	**0.7746**	0.7571
bottle	0.3305	0.5136	0.6342	0.7011	0.6821	0.6962	0.7168	0.7157	**0.7269**	**0.7170**
bus	0.6397	0.7902	0.8424	0.8679	0.8799	0.8826	0.8764	**0.9055**	**0.8896**	0.8895
car	0.8282	0.8679	0.9046	0.9197	0.9315	0.9278	0.9289	**0.9415**	0.9361	**0.9444**
cat	0.6149	0.6459	0.7050	0.7604	0.7967	0.8057	0.8478	0.8442	**0.8677**	**0.8678**
chair	0.2801	0.4658	0.5293	0.5943	0.6065	0.6606	0.6536	0.6804	**0.6816**	**0.6644**
cow	0.4893	0.5889	0.7511	0.7957	0.8564	0.8563	0.8770	**0.8923**	**0.8962**	0.8879
diningtable	0.4040	0.6601	0.7300	0.7513	0.7683	0.7757	**0.8127**	0.7853	0.7917	**0.8261**
dog	0.4538	0.5576	0.6240	0.7286	0.7677	0.8106	0.8141	0.8312	**0.8455**	**0.8423**
horse	0.6824	0.7171	0.8435	0.8705	0.8950	0.9124	**0.9176**	**0.9177**	0.9106	0.9152
motorbike	0.6550	0.7600	0.8192	0.8680	0.8879	0.9018	0.8965	**0.9083**	**0.9223**	0.9071
person	0.7515	0.8043	0.8317	0.8594	0.8697	0.8719	0.8838	0.8833	**0.8884**	**0.8909**
pottedplant	0.1345	0.3226	0.4444	0.5016	0.5002	0.5580	0.5474	**0.5797**	**0.5766**	0.5711
sheep	0.4641	0.6195	0.7189	0.7701	0.8222	0.8462	0.8314	**0.8604**	**0.8470**	0.8432
sofa	0.5112	0.6142	0.6920	0.7316	0.7497	0.7729	0.7776	0.7914	**0.7973**	**0.8028**
train	0.6819	0.7584	0.8198	0.8357	0.8502	0.8604	**0.8765**	0.8655	0.8708	**0.8788**
tvmonitor	0.5894	0.6781	0.7464	0.7856	0.7694	**0.8276**	0.8300	**0.8346**	0.8332	**0.8469**
Mean AP50	0.5344	0.6482	0.7265	0.7730	0.7897	0.8140	0.8206	0.8315	**0.8337**	**0.8361**

Figure 6.6: The master test set AP50 score for each class. The highest and second-highest AP50 scores are bolded.

	Training round																			
Object	1		2		3		4		5		6		7		8		9		10	
aeroplane	118	3.8%	174	2.8%	278	2.9%	382	3.1%	474	3.1%	555	3.1%	620	3.1%	683	3.1%	736	3.2%	916	3.6%
bicycle	105	3.4%	186	2.9%	285	3.0%	398	3.2%	496	3.2%	592	3.3%	679	3.4%	754	3.4%	794	3.5%	826	3.2%
bird	145	4.7%	243	3.9%	341	3.6%	424	3.4%	510	3.3%	609	3.4%	696	3.4%	787	3.6%	855	3.7%	1106	4.3%
boat	64	2.1%	179	2.8%	275	2.9%	360	2.9%	447	2.9%	543	3.0%	628	3.1%	787	3.6%	692	3.0%	704	2.7%
bottle	121	3.9%	281	4.5%	434	4.6%	580	4.6%	696	4.5%	814	4.5%	896	4.4%	960	4.4%	993	4.3%	1030	4.0%
bus	93	3.0%	190	3.0%	290	3.1%	397	3.2%	490	3.2%	583	3.2%	612	3.0%	620	2.8%	621	2.7%	627	2.4%
car	260	8.5%	490	7.8%	706	7.5%	934	7.5%	1144	7.4%	1309	7.2%	1456	7.2%	1574	7.2%	1677	7.3%	1990	7.8%
cat	196	6.4%	284	4.5%	385	4.1%	495	4.0%	616	4.0%	717	4.0%	816	4.0%	916	4.2%	992	4.3%	1428	5.6%
chair	237	7.7%	517	8.2%	825	8.7%	1084	8.7%	1299	8.4%	1501	8.3%	1637	8.1%	1740	7.9%	1801	7.9%	1870	7.3%
cow	65	2.1%	141	2.2%	200	2.1%	279	2.2%	351	2.3%	434	2.4%	449	2.2%	451	2.1%	453	2.0%	455	1.8%
diningtable	112	3.7%	305	4.8%	477	5.1%	611	4.9%	723	4.7%	836	4.6%	871	4.3%	885	4.0%	891	3.9%	904	3.5%
dog	189	6.2%	371	5.9%	515	5.5%	676	5.4%	832	5.4%	979	5.4%	1116	5.5%	1228	5.6%	1334	5.8%	1727	6.7%
horse	88	2.9%	176	2.8%	262	2.8%	354	2.8%	456	3.0%	549	3.0%	640	3.2%	722	3.3%	749	3.3%	777	3.0%
motorbike	85	2.8%	183	2.9%	278	2.9%	372	3.0%	465	3.0%	548	3.0%	640	3.2%	711	3.2%	754	3.3%	785	3.1%
person	739	24.1%	1492	23.7%	2241	23.7%	2984	23.9%	3712	24.1%	4406	24.3%	4953	24.5%	5402	24.7%	5643	24.7%	6469	25.2%
pottedplant	90	2.9%	245	3.9%	398	4.2%	508	4.1%	604	3.9%	694	3.8%	771	3.8%	823	3.8%	828	3.6%	841	3.3%
sheep	53	1.7%	123	2.0%	181	1.9%	246	2.0%	330	2.1%	391	2.2%	417	2.1%	420	1.9%	422	1.8%	423	1.7%
sofa	113	3.8%	287	4.6%	415	4.4%	550	4.4%	670	4.4%	795	4.4%	914	4.5%	987	4.5%	1009	4.4%	1067	4.2%
train	102	3.3%	202	3.2%	289	3.1%	370	3.0%	464	3.0%	556	3.1%	640	3.2%	714	3.3%	774	3.4%	813	3.2%
tvmonitor	90	2.9%	237	3.8%	362	3.8%	488	3.9%	594	3.9%	694	3.8%	783	3.9%	852	3.9%	863	3.8%	874	3.4%
Total objects	3066	100%	6306	100%	9437	100%	12492	100%	15373	100%	18105	100%	20234	100%	21911	100%	22881	100%	25632	100%
Total images	2000		3816		5638		7453		9214		10922		12322		13473		14217		16551	

Figure 6.7: Number of objects in combination of train, validation, and test sets during the training rounds.

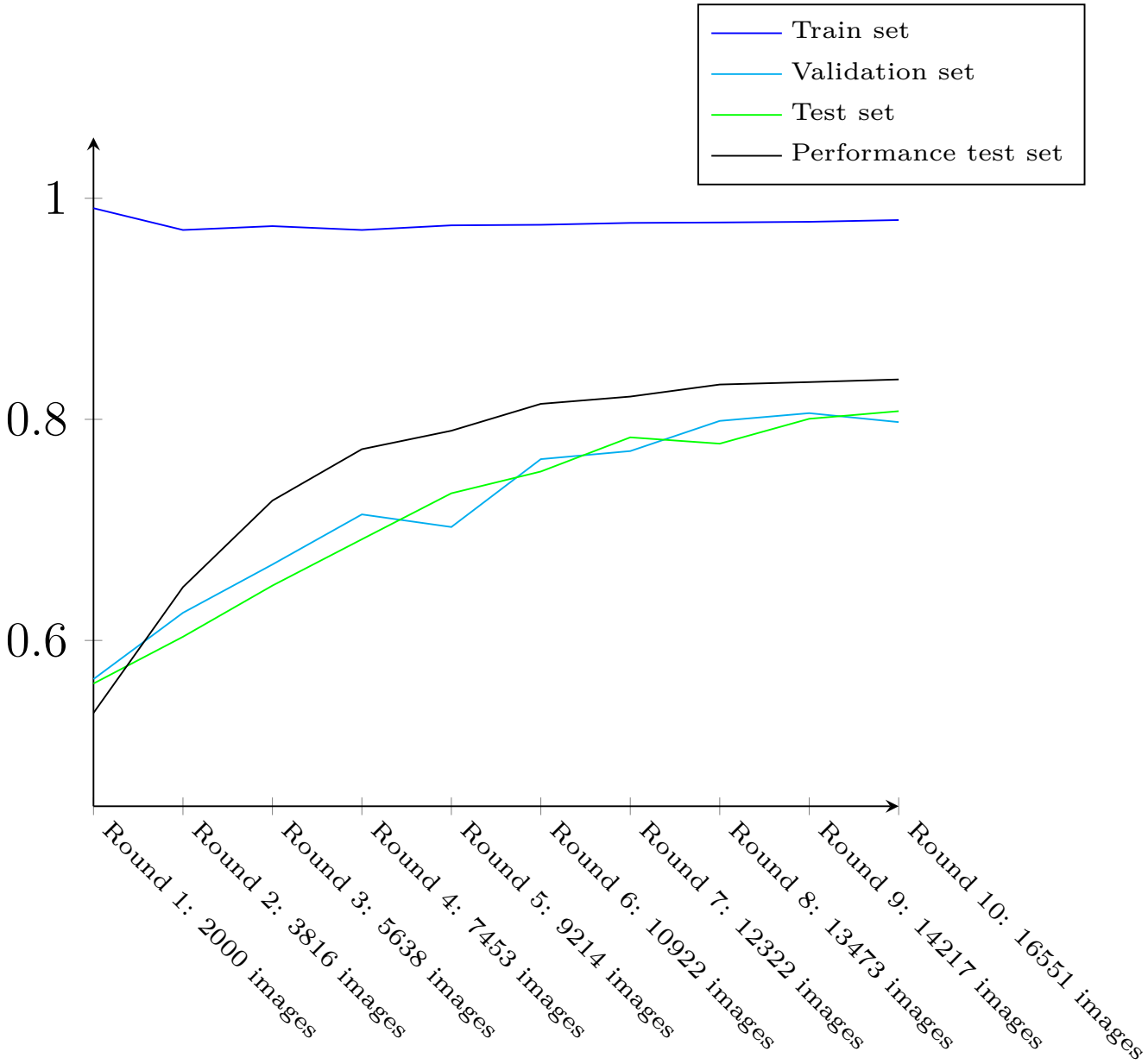

Figure 6.8: AP50 score evolution through the training rounds.

7. Conclusions

In this thesis, our main topic was to study active learning to minimize the amount of training data required to train a modern, state-of-the-art object detector that predicts the estimated localization IoU. We wanted to make the proposed method useful in real-life use cases, and that's why it needed to meet the following requirements:

1. The proposed method needs to be computationally efficient.
2. The proposed sampling method must keep the training dataset balanced (sample images from each object class).

As described before, our proposed method fulfills these requirements. The experiment shows that it can be used efficiently to sample the most informative images and save a vast amount of data labeling work. Also, we are the first ones to leverage the predicted IoU in the active learning process.

In the beginning, we studied different object detection architectures to find the most suitable one for our research. Based on this research, we decided to focus on the YOLOX, which was published in August 2021 [11]. There were four main reasons why we selected to use YOLOX. First, it is the most accurate real-time object detector, and that's why the best candidate for most real-life object detection applications. Second, it uses three interesting properties that have not been used before in active learning.

From our point of view, the most interesting properties in YOLOX are mixup data augmentation, IoU prediction, and the decoupled head. Each of these properties affects how the active learning method should be built. First, the mixup creates linear behavior between the decision boundaries, which affects predicted detection confidences. Second, with the decoupled head, the classification and localization have their own heads and last convolution layers, which makes it more important to estimate image informativity from both perspectives. As a solution, we have proved that the predicted IoU can be used to estimate image informativity from the localization perspective.

At the thesis's beginning, we went through the typical object detector architecture and the most important building blocks. Then we introduced the augmentation techniques to train these models. Both of these topics have developed quickly dur-

ing the last five years, and the YOLOX combines the latest techniques from several research papers.

Then we introduced the concept of active learning. The main principles of active learning have stayed relatively similar during the last 15 years, and one of our active learning sources was from 2007, which we found to be still relevant and applicable to the latest deep learning-based models. There was only a limited amount of new research available on this topic, which is surprising when we consider the potential benefits that can be achieved with active learning.

Before going to the experiment, we introduced the YOLOX object detector. First, we went through the whole YOLO family object detectors development timeline and then focused on the YOLOX object detector. During this process, we found that the improvement between two sequential models is usually relatively small, and updates typically come from the other research papers. But when we compare the first and the latest model, then there is a significant difference.

The last chapter of the thesis was the experiment. In the experiment, we proved that our proposed method makes the labeling process more efficient, and it can save a significant amount of annotation time. However, we expected even better results from our method, and one reason may be the used dataset. According to Haussmann et al., typical object detection datasets are already cleaned to contain mostly informative samples [14]. This would cause the active learning method to be less beneficial than in a real-life situation.

It is likely that our method could be improved few ways. First, we did not use any sophisticated method to define the seed set, which may lower the benefits. Because the most informative images are sampled with the object detector itself, it is likely the model trained with better seed sets would have sampled different images. Second, our way of defining the informativity image score is may not be optimal. In our method, we multiple the classification confidence and the predicted IoU to get the final detection confidence score. When defining image informativity, better results might be achievable by using these two detection confidence numbers separately. Also, when defining the classification confidence, we only used the highest classification confidence, but YOLOX predicts confidence for each object class for each detected object. This means that there can be more than one high confidence classification class for a single detected object, and using this information could possibly give even better results.

Bibliography

[1] Model zoo. https://modelzoo.co/. Accessed: 2021-12-06.

[2] Pytorch hub. https://pytorch.org/hub/. Accessed: 2021-12-06.

[3] Tensorflow hub. https://tfhub.dev/. Accessed: 2021-12-06.

[4] Y. Amit, P. Felzenszwalb, and R. Girshick. *Object Detection*, pages 1–9. Springer International Publishing, Cham, 2020.

[5] A. Bochkovskiy, C. Wang, and H. M. Liao. YOLOv4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020.

[6] C. Brust, C. Käding, and J. Denzler. Active learning for deep object detection. *CoRR*, abs/1809.09875, 2018.

[7] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, pages 213–229, Cham, 2020. Springer International Publishing.

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[9] M. Everingham and J. Winn. The pascal visual object classes challenge 2012 (VOC2012) development kit. *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep*, 8:5, 2011.

[10] Z. Ge, S. Liu, Z. Li, O. Yoshie, and J. Sun. OTA: Optimal transport assignment for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 303–312, 2021.

[11] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun. YOLOX: exceeding YOLO series in 2021. *CoRR*, abs/2107.08430, 2021.

[12] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[13] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi. IQA: visual question answering in interactive environments. *CoRR*, abs/1712.03316, 2017.

[14] E. Haussmann, M. Fenzi, K. Chitta, J. Ivanecky, H. Xu, D. Roy, A. Mittel, N. Koumchatzky, C. Farabet, and J. M. Alvarez. Scalable active learning for object detection. *CoRR*, abs/2004.04699, 2020.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 346–361, Cham, 2014. Springer International Publishing.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[17] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[18] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu. A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868, 2019.

[19] G. Jocher, A. Stoken, A. Chaurasia, J. Borovec, NanoCode012, TaoXie, Y. Kwon, K. Michael, L. Changyu, J. Fang, A. V, Laughing, tkianai, yxNONG, P. Skalski, A. Hogan, J. Nadar, imyhxy, L. Mammana, AlexWang1900, C. Fati, D. Montes, J. Hajek, L. Diaconu, M. T. Minh, Marc, albinxavi, fatih, oleg, and wanghaoyang0106. YOLOv5. https://github.com/ultralytics/yolov5, 2020.

[20] C. Kao, T. Lee, P. Sen, and M. Liu. Localization-aware active learning for object detection. *CoRR*, abs/1801.05124, 2018.

[21] K. Kim and H. S. Lee. Probabilistic anchor assignment with iou prediction for object detection. In *European Conference on Computer Vision*, pages 355–371. Springer, 2020.

[22] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[23] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[24] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[25] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[26] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[27] I. Muslea, S. Minton, and C. A. Knoblock. Selective sampling with redundant views. In *AAAI/IAAI*, pages 621–626, 2000.

[28] E. S. Olivas, J. D. M. Guerrero, M. M. Sober, J. R. M. Benedito, and A. J. S. Lopez. *Handbook Of Research On Machine Learning Applications and Trends: Algorithms, Methods and Techniques - 2 Volumes*. Information Science Reference - Imprint of: IGI Publishing, Hershey, PA, 2009.

[29] F. Olsson. *Bootstrapping Named Entity Annotation by Means of Active Machine Learning: A Method for Creating Corpora*. PhD thesis, , SICS, 2008.

[30] F. Olsson. A literature survey of active machine learning in the context of natural language processing. Technical Report 2009:06, SICS, 2009.

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

[32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[33] J. Redmon and A. Farhadi. YOLO9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[34] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[35] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2017.

[36] O. Russakovsky, L.-J. Li, and L. Fei-Fei. Best of both worlds: Human-machine collaboration for object annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[37] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[38] G. Song, Y. Liu, and X. Wang. Revisiting the sibling head in object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11563–11572, 2020.

[39] H. Su, J. Deng, and L. Fei-Fei. Crowdsourcing annotations for visual object detection. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[40] Z. Tian, C. Shen, H. Chen, and T. He. FCOS: Fully Convolutional One-Stage Object Detection. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9626–9635, 2019.

[41] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh. CSPNet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[42] L. Wang, M. Han, X. Li, N. Zhang, and H. Cheng. Review of classification methods on unbalanced data sets. *IEEE Access*, 9:64606–64628, 2021.

[43] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[44] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017.

[45] S. Zhang, L. Song, S. Liu, Z. Ge, Z. Li, X. He, and J. Sun. Workshop on autonomous driving at CVPR 2021: Technical report for streaming perception challenge. *CoRR*, abs/2108.04230, 2021.

[46] Y. Zhang, P. Sun, Y. Jiang, D. Yu, Z. Yuan, P. Luo, W. Liu, and X. Wang. ByteTrack: Multi-object tracking by associating every detection box. *CoRR*, abs/2110.06864, 2021.

[47] 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*, 30(11):3212–3232, 2019.

[48] J. Zhu, H. Wang, and E. Hovy. Learning a stopping criterion for active learning for word sense disambiguation and text classification. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*, 2008.

[49] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2021.