Tampere University

Samuli Pohjola

OBJECT DETECTOR FINE-TUNING FOR COMPUTER VISION APPLICATIONS

In surveillance and adverse condition data domains

Master of Science Thesis
Faculty of Information Technology and Communication Sciences
Examiners: Joni Kämäräinen
Esa Rahtu
May 2022

ABSTRACT

Samuli Pohjola: Object Detector Fine-tuning for Computer Vision Applications
Master of Science Thesis
Tampere University
Master's Degree Programme in Information Technology
May 2022

The advancements of CNNs have made object detection increasingly popular and now it is used in a wide range of applications. There are many pretrained object detector implementations available, but these models do not always fulfil the needs of real applications. That is why retraining for application needs is often required although it can be expensive. One efficient way of retraining is fine-tuning a pretrained model with a small number of images.

In this thesis object detection fine-tuning is studied to estimate the detection accuracy improvements gained by fine-tuning with different data quantities. Experimentation is done on publicly available data from surveillance and adverse conditions data domains with the YOLOv5 object detector. The experimental results show that 50 to 75 fine-tuning images are enough to produce distinctively improved results for a single data source. Having more images would only increase the accuracy to a small extent, but also having less than 30 would produce worse results than the pretrained model. Similar results are obtained when fine-tuning a more general model with 15 images from each source. The experiments also show that the training data needs to have enough object instances for successful learning.

Additionally, an automatic fine-tuning image selection algorithm is proposed in the thesis. It uses the pretrained detection model to algorithmically select fine-tuning images. In the experiments the algorithmic selection is seen to produce slightly better accuracy and less deviation in results when compared to random selection. The accuracy improvements are marginal, but the algorithm is useful with real-life camera streams where a suitable time for random selection is hard to estimate. In these cases, the algorithm could speed up data selection for fine-tuning and that way reduce expenses.

Keywords: Deep learning, Object detection, fine-tuning, CNN, YOLOv5

TIIVISTELMÄ

Samuli Pohjola: Kohteentunnistimen hienosäätö konenäkösovelluksissa
Diplomityö
Tampereen yliopisto
Tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Toukokuu 2022

Konvoluutioon perustuvien neuroverkkojen kehitys on tehnyt kohteen tunnistuksesta suosittua ja nykyään sitä hyödynnetään monissa erilaisissa sovelluksissa. Kohteen tunnistukseen on tarjolla monia valmiiksi koulutettuja malleja, mutta nämä mallit eivät aina vastaa oikeiden sovelluskohteiden vaatimuksia. Sen vuoksi tunnistin täytyy usein kouluttaa uudelleen vastaamaan sovelluskohteen tarpeita, vaikka uudelleenkouluttaminen on hidas ja työläs prosessi. Yksi tapa tehdä uudelleenkoulutus tehokkaasti on hienosäätää valmista mallia pienellä määrällä kuvia.

Tässä työssä tutkitaan kohteentunnistimen hienosäätöä, jotta saataisiin arvio kuvamäärästä, joka tarvitaan paremman tarkkuuden saavuttamiseksi. Työssä tehdään hienosäätökokeita YOLOv5-kohteentunnistimella. Kohteentunnistimen kouluttamiseen testeissä käytetään julkisia kuvasarjoja valvontakameroista ja tunnistukselle haitallisista sääolosuhteista. Kokeiden tulokset osoittavat, että 50–75 kuvaa on riittävä määrä tuottamaan hyviä tuloksia yhden kameran tai olosuhteen kuvia käytettäessä. Kuvien lisääminen parantaa tarkkuutta vain vähän ja kuvien vähentäminen alle 30:n tuottaa huonompia tuloksia kuin valmiiksi koulutettu malli. Kun haetaan yleisempää mallia, samanlaisia tuloksia saadaan käyttämällä 15 kuvaa jokaisesta kamerasta tai olosuhteesta, jossa mallin halutaan toimivan. Kokeissa myös selviää, että koulutusdatassa tulee olla riittävä määrä tunnistettavia kohteita, jotta niiden oppiminen onnistuu.

Lisäksi työssä esitellään algoritmi, jolla voidaan automaattisesti valita kuvia kohteen tunnistimen hienosäätöä varten. Algoritmi perustuu valmiiksi koulutetun kohteentunnistimen tekemiin tunnistuksiin, joiden pohjalta algoritminen valinta tehdään. Kokeissa algoritminen valinta tuottaa hieman paremman tarkkuuden ja vähemmän hajontaa tuloksissa kuin satunnainen valinta. Algoritmilla saavutettu parannus on kuitenkin marginaalista, mutta algoritmi nähdään hyödylliseksi käytettäessä oikeiden kameroiden kuvavirtoja, joissa on vaikea arvioida sopivaa aikaa satunnaiselle valinnalle. Näissä tapauksissa algoritmi nopeuttaa hienosäätökuvien keräämistä ja siten vähentää siitä koituvia kustannuksia.

Avainsanat: Syväoppiminen, Kohteen tunnistus, hienosäätö, konvolutiivinen neuroverkko, YOLOv5

PREFACE

When I first started my studies on the information technology field of study in 2017, I did not know programming at all. At that time, I was more interested in the structure of computer components, and I thought I would study more digital and computer technology. But as it happens, I ended up studying all kinds of programming, embedded and digital systems, signal processing and machine learning and eventually found my calling on the field of computer vision. It has been an eventful 5-year journey, but it is finally coming to an end with the completion of this thesis.

Before the end of this journey they call student life I would like thank Tampere University and especially the former Tampere University of Technology for the interesting and high-quality teaching. Special thanks go to Joni Kämäräinen whose advice and suggestions tremendously helped me to complete this thesis. I will long remember our fruitful and interesting discussions which always helped to push the work onwards.

I would also like to thank Wapice for support and a possibility to conduct this thesis work for them. I want to thank Mickey Shroff for giving me this opportunity and for interesting ideas and possibilities for this thesis work. I also want to express my thanks to my colleagues for their help and expert advice on computer vision research.

Additionally, I want to show my gratitude to my friends and fellow students who helped me to get through this process by giving me peer support and by helping me to relax in my free time. Special thanks go to Vili Kautto and Heikki Lahtinen for giving improvement suggestions and for proof-reading the thesis.

Last but definitely not the least I want to express my deepest gratitude for my family who helped and supported me through this thesis and through the whole university studies. I want to thank my sister for her invaluable help during this process' most difficult times. Also, thanks to my parents for their gentle but determined support which helped me to push through this thesis to graduate in time and to get a good start for my future life.

Tampere, 8 May 2022

Samuli Pohjola

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AAU	Aalborg University
AUC	Area under curve
AP	Average precision
CADCD	Canadian Adverse Driving Conditions dataset
CNN	Convolutional neural network
COCO	Common Objects in Context
CSPDarknet53	Cross stage partial Darknet53
DAWN	Detection in Adverse Weather Nature
DETR	Detection Transformer
DNN	Deep neural network
DPM	Deformable part-based model
GDPR	General Data Protection Regulation
HD	High-definition
HOG	Histogram of oriented gradients
IoU	Intersection over union
mAP	Mean average precision
NLP	Natural language processing
NMS	Non-maximum suppression
ONNX	Open Neural Network Exchange
PAN	Path aggregation network
RCNN	Region-based Convolutional Neural Network
RGB	Red green blue color model
SPP	Spatial pyramid pooling
SSD	Single shot detector
YOLO	You only look once

F_n	number of false negatives
F_p	number of false positives
T_n	number of true negatives
T_p	number of true positives

1. INTRODUCTION

CNNs (Convolutional neural networks) have dominated the object detection field for some time now [1]. They first rose to prominence in 2012 with the introduction of AlexNet and have continued to develop since then [2]. The evolution of CNNs has hugely improved the performance of computer vision with various architectures like R-CNN (Region-based Convolutional Neural Network), Fast R-CNN and YOLO (You only look once) [3]. Nowadays they still hold a significant role in object detection, although there are also other prominent technologies like Transformer based object detectors. Although transformer-based architectures have been shown to produce good results in computer vision [4] their deployment options are still limited and they are not as easy to use as competing deep CNN architectures.

The development of CNN based object detection has resulted in architectures that can provide accurate predictions with short inference times. There are premade and pretrained implementations of such CNN detectors that are fast enough to be used in real-time object detection even with modest hardware [5, 6]. The possibility of real-time object detection has opened many new increasingly interesting real-life applications for computer vision. Real-time object detection enables the usage of object trackers or real-time event detection, which can be used in, for example, security or smart city data analytics. Unfortunately, these use cases often have different view angles, detection needs and background conditions compared to what the pretrained models offer, which raises the need for detection model retraining.

The need to retrain emphasizes the central issue of CNN-based object detection. Training CNN detectors often needs large amounts of accurately labeled data to produce good results [7, 8]. To accomplish this, data needs to be collected and annotated which can be time-consuming and expensive. To lower these expenses, crowdsourcing can be used to annotate the data and it has been shown that this can even yield accurate results in some scenarios [9]. However, in security and smart city applications the used data often includes private or confidential information and crowdsourcing the data could be a violation of regulations like the GDPR (General Data Protection Regulation).

Another solution is using a pretrained detection model as a basis and fine-tune it to correspond to the needs of the specific task. The process of fine-tuning makes use of an

existing network by starting the training process with weights that have been pretrained with a large-scale dataset and continues to fine-tune these weights with the new data [10]. This process is depicted in Figure 1. The features learned by the weights optimized by thousands of training images will remain during fine-tuning, but their relation to new or old classes can be fine-tuned. In fine-tuning some layers of the model can also be randomly initialized or frozen to the pretrained values. Random initialization is needed for the output layer of the network if new classes are added in fine-tune training. It can still be useful even when the detected classes stay the same.

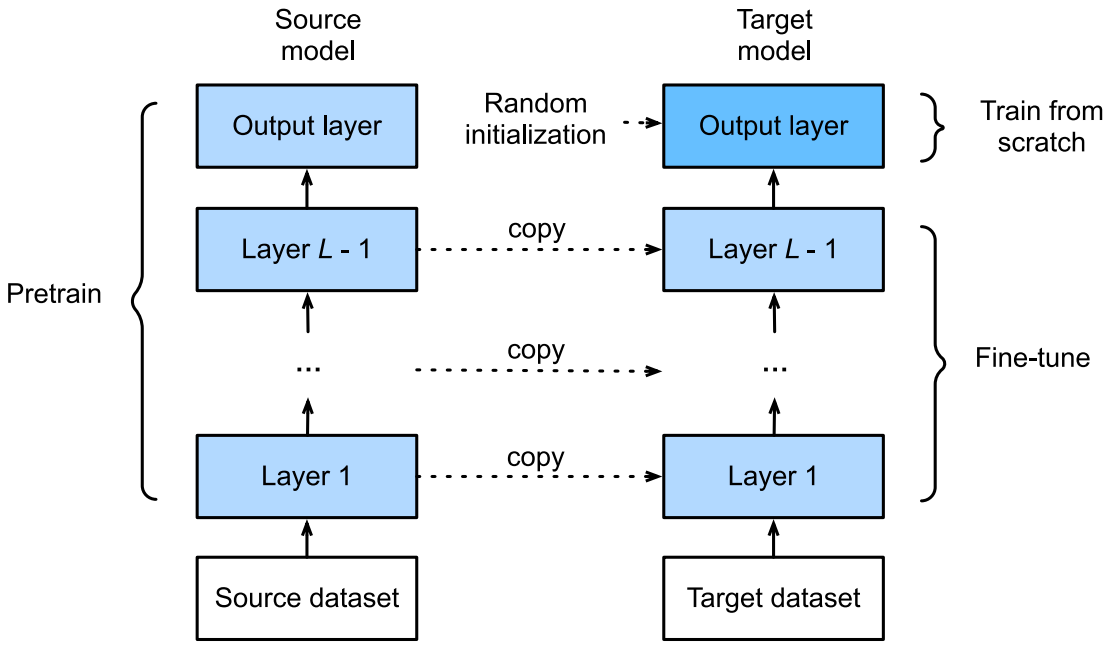

Figure 1. *Neural network fine-tuning uses pretrained model weights as a basis for new model training [11].*

It has been shown that fine-tuning requires less data than complete retraining [12]. Fine-tuning even with a small amount of data can produce significant accuracy gains. An example is presented in Figure 2, where ground truth and predictions with a pretrained model and a fine-tuned detector are shown. Because of the costs and data limitations, fine-tuning a pretrained model with self-annotated data is usually the best solution for scenarios with sensitive data. However, the question of how much data and what quality is needed to produce good fine-tuning results remains open.

Figure 2. *Images showing benefits of small-scale fine-tuning. The leftmost images depict ground truth. The images in the middle and on the right show predictions from a pretrained model but for the rightmost images the model has been fine-tuned with 50 images.*

The research objective of this thesis is to find out what amount of data is needed for fine-tuning a pretrained object detection model to achieve improved accuracy. This objective is considered with real-life smart city analytics or other surveillance applications in mind. Answers to this question are searched by running fine-tuning experiments with real data and state-of-the-art CNN object detector YOLOv5. The data domains of this work's experiments are surveillance camera and adverse weather conditions, which could be both present in a smart city data analytics scenario. Publicly available data is studied and reviewed to find out the best matches for the chosen data domains.

Extensive fine-tuning experiments are conducted on the data to find out how fine-tuning data quantity effects the accuracy of the results. The goal is to find out the accuracy gains of adding more data and to deduce estimates on optimized amounts of fine-tuning data required to produce improved accuracy. Also, an automatic fine-tuning image selection algorithm is proposed and tested for easing the selection of fine-tuning data. The effects of fine-tuning data quality are also observed in the experiments. Additionally, experimentation is done with the generalization ability of the fine-tuned detection models inside the trained data domain.

The rest of the thesis' structure is as follows. Chapter 2 goes through related work on object detectors and examines datasets that depict surveillance or adverse condition data domains. Chapter 3 describes the methods and metrics used in the work's experiments and their theoretical background. Chapter 4 presents the experiments and their results in detail. Discussion on results and conclusions are also presented in Chapter 4. All results and conclusions are gathered in chapter 5 at the end of the thesis.

2. RELATED WORK

This chapter goes through research in the field of object detection and a number of public datasets that can be used for object detector training. Section 2.1 introduces the current state of object detection and technologies used in the research field. Section 2.2 examines publicly available datasets with surveillance camera view angles or adverse weather conditions. Also, the suitability of these datasets for object detector training is assessed.

2.1 Object Detection

One of the most interesting and highly studied topics in computer vision is object detection [13]. The idea of object detection is to separate interesting objects from the image background and then assign these objects a predefined class label [13–15]. The objects are often marked by drawing bounding boxes around them in the image to show their size. Object detection has been done with different methods like HOG (histogram of oriented gradients) or DPM (deformable part-based model), but the most prominent method used today is deep learning [16]. Deep learning mostly refers to DNNs (deep neural networks), which have complex architectures and many layers, and therefore capacity to learn more complex features [15].

In the present, the area of object detection can be divided into three prevalent algorithm categories: two-stage algorithms, one-stage algorithms and transformers [2]. Two-stage algorithms divide the detection task into two phases. The first phase generates region proposals on possibly interesting objects and the second phase classifies these regions [17]. Examples of two-stage detectors are R-CNN [18], SPP-Net [19], Fast R-CNN [20] and Faster R-CNN [21]. The advantage of two-stage detectors is usually accuracy at the cost of speed.

One-stage algorithms on the other hand do not use region proposals, but instead extract direct features from input images and then perform classification and localization predictions according to those features [22]. This makes the detection process faster but also lowers the accuracy when comparing to the two-stage algorithms. Popular one-stage detectors are YOLO [23], SSD (single shot detector) [24], Retina-Net [25] and CenterNet [26].

An emerging topic in computer vision is the use of transformers in visual tasks [27]. Transformers were adapted from NLP (natural language processing) to reduce structural

complexity for increased training efficiency and scalability [2]. The benefits of transformers are self-attention layers which can aggregate information from whole input sequences allowing the creation of structures with less need for hand-designed components [28]. The most well-known object detection transformer is DETR (Detection Transformer), which uses a CNN backbone with a transformer head [29]. Another popular transformer structure in object detection is the Swin Transformer that has been shown to produce state-of-the-art level detection accuracy [30–32].

2.2 Datasets

Nowadays there are a lot of publicly available datasets that can be used to train and test object detectors. These datasets often have different characteristics. They differ in data quantity and annotation quality. Choosing the right datasets to be used in research can be a difficult and time-consuming task [33]. For object detection in surveillance or adverse weather condition data domains there exist a few representative datasets, which are introduced here.

2.2.1 VIRAT

The VIRAT video dataset contains 8.5 hours of HD (high-definition) quality video clips that have been taken from cameras placed in surveillance camera angles. The dataset contains diverse scenes from mostly urban areas where human activity occurs. The videos have been shot mostly during daytime and the weather conditions differ from clear skies to light rain. [34]

The VIRAT dataset is designed to be used in action or event detection with video data, and therefore it offers localized annotations for frames that include these actions or events. In addition, bounding boxes and classes are given for objects participating in these actions. Unfortunately, the bounding boxes do not cover all objects of the described classes. This makes the dataset unsuitable for object detector training or fine-tuning, as missing class labels for clearly visible objects hamper the learning of an object detection model. An example image from the dataset is shown in Figure 3.

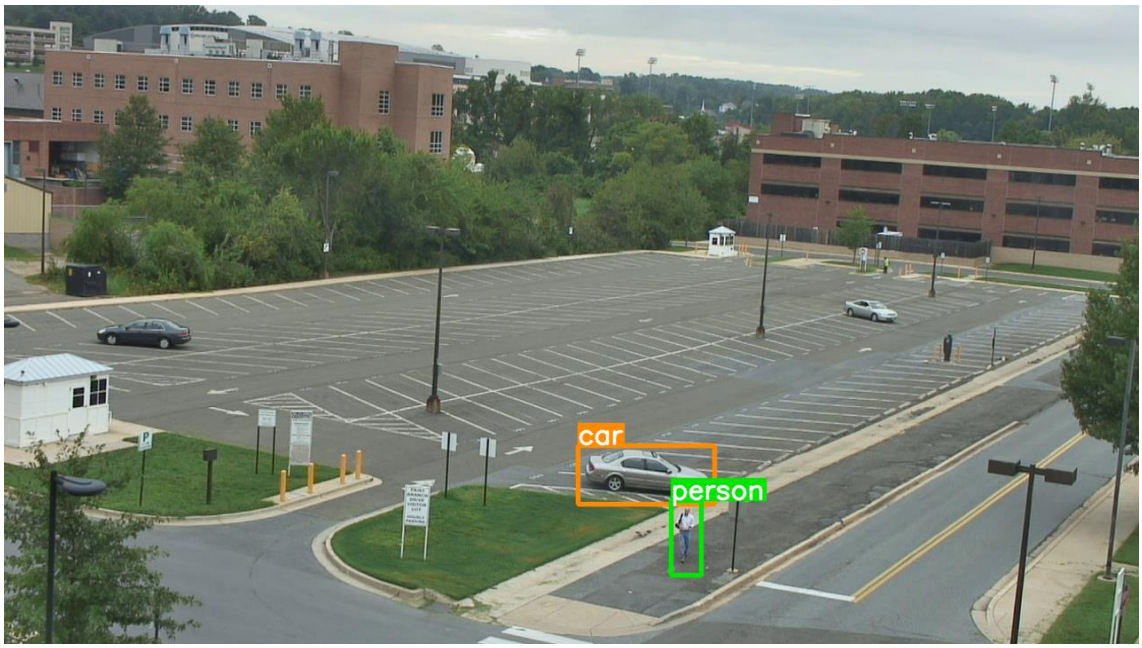

Figure 3. *An example image from VIRAT video dataset that shows the view angle and that annotations are only given for some objects in the videos.*

2.2.2 AU-DETRAC

The AU-DETRAC dataset is created from 10 hours of video data from multiple locations depicting traffic in urban settings or on highways from a surveillance camera view. The dataset is meant to be used as an object detection and tracking benchmark. The data is presented as annotated frames with 960x540 resolution that have been taken from the original videos. The weather conditions in the frames feature sunshine, clouds and some rain. Most of the frames are from daytime videos but some show a night-time scenery. [35]

The dataset's annotations are provided as bounding boxes, object identifiers and class labels for large and medium-sized vehicles including passenger cars, buses, trucks and vans. The annotations are only given for vehicles that are on the central roads of the camera view and not far away. For other parts of the frames the dataset provides static rectangular areas where objects are ignored and therefore not labelled. By hiding these ignored areas, the frames can be used in object detection training. However, the ignored areas are not perfect and in some frames partially visible vehicles do not have annotations or their bounding boxes extend far into the ignored areas. Additionally, the cameras used to take the videos are unstable and they wobble slightly which moves the ignored areas and sometimes reveals unlabelled vehicles. Example images from the dataset are presented in Figure 4.

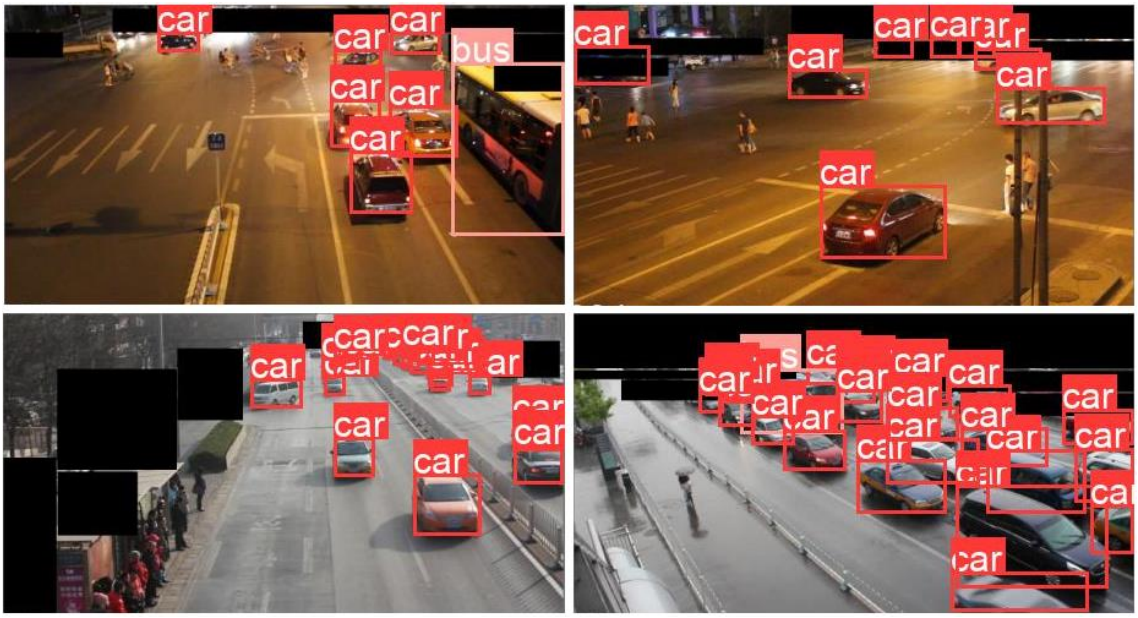

__Figure 4.__ Example images from AU-DETRAC dataset with annotations showing scene variety and problems with ignored areas. The ignored areas are marked with black rectangles.

2.2.3 CADCD

CADCD or Canadian Adverse Driving Conditions dataset is an autonomous driving dataset with images, LiDAR sweeps and other driving related data from snowy weather conditions. The data has been gathered by driving around urban areas with an autonomous vehicle platform which has produced videos, LiDAR sweeps and other information. The weather in the data varies from light snowfall to heavy snowfall and all of the data has been collected during daytime. [36]

The published dataset contains labelled traffic scenes that feature camera images and LiDAR points from the data gathering vehicle's sensors and 3D-bounding boxes for objects in the images. The annotation data has been given in a format that allows reconstruction of these scenes in 3D, showing the locations and orientations of other objects near the data gathering vehicle. This allows for in-depth analysis of the driving scenes, but unfortunately is not as useful for object detection. The bounding boxes are given in 3D coordinates and can be projected to the provided camera images as 2D bounding boxes, but this transition causes some errors in the box precision. The main problem for object detector training is that the annotations do not specify if an object is visible in the camera image or not. This leads to a large number of bounding boxes without visible objects. An example of these empty bounding boxes and a reconstructed LiDAR scene is shown in Figure 5.

Figure 5. *Image and LiDAR scene from CADCD. The image shows multiple bounding boxes in the nearby building's wall. This is because the annotation data does not mention object visibility in camera images. Adapted from [28].*

2.2.4 DAWN

Detection in Adverse Weather Nature or DAWN is an object detection dataset that depicts traffic in various adverse weather conditions in urban and highway scenery. The data is presented as a collection of around 1000 images with a wide range of different resolutions and varying view angles. The images have been divided into 4 different weather condition categories: fog, rain, snow and sand. The weather in these categories varies between light mist, heavy fog, light rain, rainstorm, light snowfall, blizzard and sandstorm. Most of the images are taken during daytime but some also feature nighttime. [37]

Annotations in the DAWN dataset are given for all road users in a format that can be easily used with many object detectors. There are some mistakes in the annotations such as wrong class labels for some objects, missing labels for visible objects or wrong sized bounding boxes, but the overall data quality is good. Figure 6 shows example images from the dataset but overrepresents the portion of mistakes in the data. This makes the dataset suitable for object detector fine-tuning, but it is too limited to be used in complete retraining. An object detection model fine-tuned with image data that represents various view angles and has more object variation will produce a more generalized model

as a result [38], but it is questionable if this dataset's 1000 images are enough to produce a good generalization.

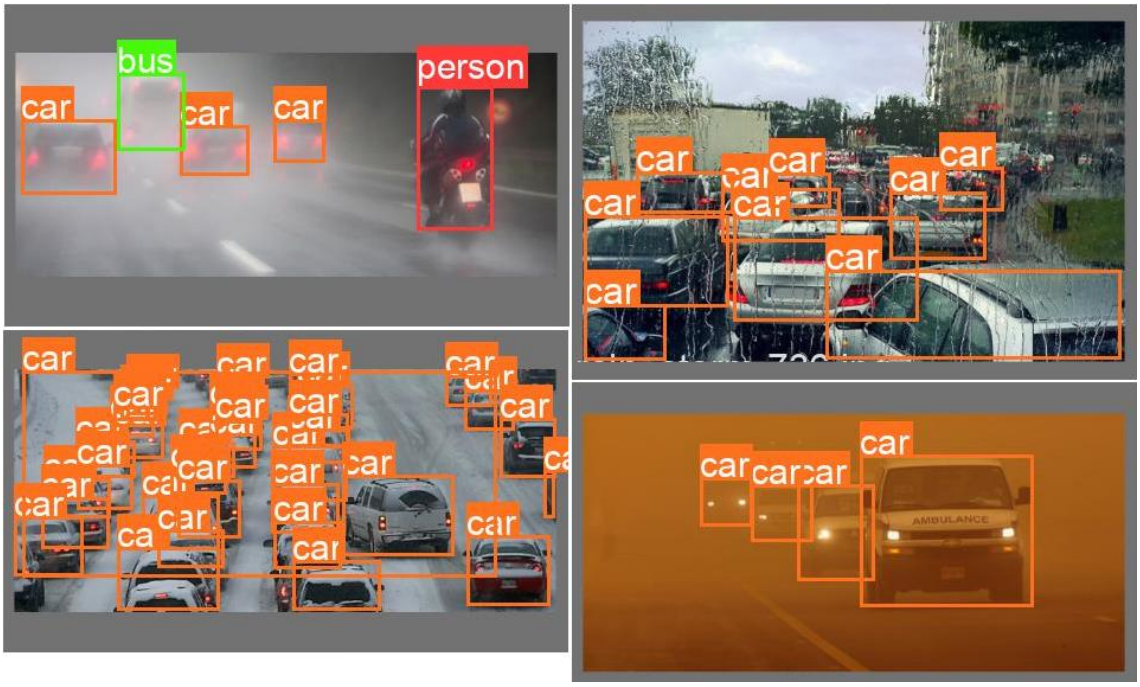

Figure 6. *Example images from DAWN dataset showing the 4 different weather condition categories: fog, rain, snow and sand. The images also show missing and oversized bounding box errors.*

2.2.5 AAU RainSnow

AAU (Aalborg University) RainSnow Traffic Surveillance Dataset provides video data from 7 different traffic surveillance cameras with RGB and thermal infrared capabilities. The original use for the dataset is rain removal algorithm evaluation with segmentation and feature tracking. There are 22 short videoclips and each of them have been split into 100 randomly selected and labelled RBG and thermal image pairs with a 640x480 resolution. Time of day in the labelled image pairs is split quite evenly between daytime and night-time. Weather conditions in the images feature varying amounts of rainfall and snowfall and also some fog and haze. [39]

The AAU RainSnow dataset's annotations are given as bounding boxes and segmentation masks for all road users that are not too far away from the camera. This means that, for example, cars parked beside the road are not annotated. To prevent these unlabelled areas from affecting training or evaluation, the dataset offers image masks that can be used to hide the areas where annotations are not provided. Unfortunately, some of the image masks do not completely cover all stationary objects on the side of the roads and some annotations are overlapping or completely inside the masked area. The cameras also wobble a bit in windy conditions, which moves the masked areas. In addition to

masking errors the dataset contains a few frames that are missing annotations for some road users or have wrong class labels on some objects. Nevertheless, these mistakes only cover a small percentage of the dataset's images and rest of the data is suitable for object detection training and fine-tuning. Figure 7 shows example images from the dataset.

Figure 7. *Example images form AAU RainSnow dataset showing variation and challenging detection tasks but also some mistakes. Black areas in the images represent the masks for unlabelled areas.*

2.2.6 MOTChallenge

MOTChallenge is a multiple object tracking benchmark meant to be used to evaluate the performance on object trackers. It provides multiple videos with varying length and resolution from urban areas with multiple objects, but mostly people moving around. Some of the videos have a surveillance camera view angle and some have been taken from ground level. The videos are taken both during daytime and night-time and the weather in the videos is always clear or sunny. [40]

Annotations in the MOTChallenge are given as tracking information and bounding boxes for the tracked objects. This tracking information is given for all moving objects in the videos and to some stationary ones. In some videos tracking annotations are given for uninteresting objects like vending machines or poles. The annotations however do not

include class labels for the tracked objects, even though they are not all of the same object class. Training an object detector model with this kind of data could hamper the models training process and result in a model that produces uninteresting data. This makes the data unsuitable for object detector training. Figure 8 shows example frames from the MOTChallenge with the given annotations.

Figure 8. *Image extracted from a MOTChallenge video with given annotations. The annotation data does not include class labels which makes the data unsuitable for object detector training.*

2.3 Summary

There are many prominent object detectors in the field of computer vision as was discussed in section 2.1. The famous YOLO algorithm is one of them as it is one of the fastest and most accurate single-stage detector structures. As detection speed is often a crucial element in real-life applications the object detector used in this thesis' experiments was chosen to be based on the YOLO algorithm.

There are many publicly available datasets that depict surveillance or adverse weather condition data domains but not all of them can be used to train or fine-tune object detectors. The AAU RainSnow and DAWN datasets were chosen to be used for the purposes of this theses' experiments as they offered usable annotations for data that represented the selected data domains. The dataset choices are discussed more in depth in section 4.1.

3. METHODS

The object detection technologies and performance metrics used in this thesis are presented in this chapter. Section 3.1 introduces the YOLOv5 object detector and goes through the methods used in its structure. Section 3.2 discusses common object detection performance metrics.

3.1 YOLOv5

Released in 2020 YOLOv5 is one of the latest iterations of the YOLO-based object detectors [41]. YOLO detector is a single-stage detector built on the CNN based YOLO algorithm. The YOLO algorithm was first introduced in 2016 as YOLOv1 [42]. It was a fast detector that performed object detection through a regression problem [43]. The input was divided into regions where preliminary bounding boxes and class categories were predicted and the most unlike ones were removed according to thresholds and the NMS (non-maximum suppression) algorithm [44]. The NMS algorithm removes overlapping bounding boxes based on a predefined IoU (intersection over union) score.

The later YOLO versions continued from this basis and introduced new features in every iteration. YOLOv2 added a new backbone with the introduction of Darknet-19. It also changed the bounding box predictions to use anchor boxes which are initial sizes for predicted boxes. YOLOv3 then updated the structure with Darknet-53 backbone and modified and increased the number of YOLOv2 anchor boxes to improve small object detection. YOLOv4 further improved the performance and accuracy of YOLOv3 by changing to CSPDarknet53 (cross stage partial Darknet53) backbone and adding specialized improvements like Mosaic data enhancement, SPP (spatial pyramid pooling), PANet (path aggregation network) and Mish activation function. [43, 44]

YOLOv5 is a continuation of the previous YOLO version and is structurally quite closely related to YOLOv4, but with some new features and changes. YOLOv5 relies on the same backbone as YOLOv4: the CSPDarknet53 with SPP but with an added Focus layer that reduces memory usage and increases forward and backward propagation [45]. The neck of YOLOv5 is also built on the basis of YOLOv4 and uses an adapted version of PANet structure with a new FPN (feature pyramid network) that helps with propagation of low-level features [46]. As the last part of the network YOLOv5 uses the same YOLO

layer head as YOLOv3 and YOLOv4, and as with all earlier versions of YOLO the resulting output is finished with the NMS algorithm. The architectural structure of YOLOv5 is shown in Figure 9.

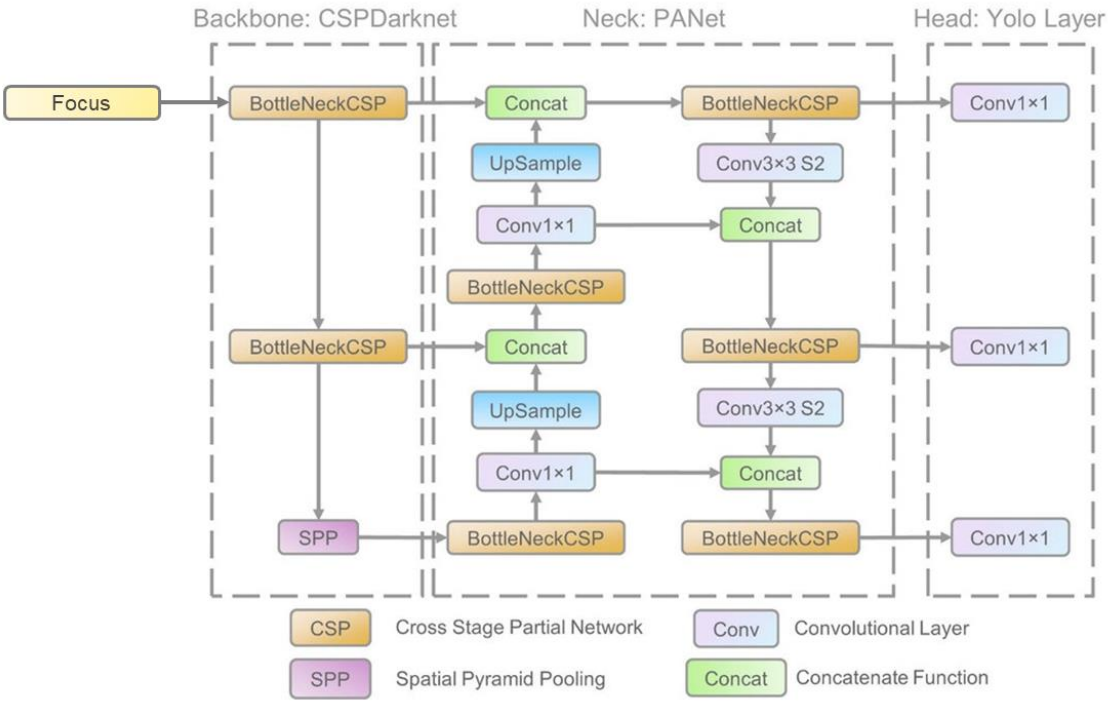

Figure 9. *YOLOv5 architecture including CSPDarknet53 backbone, PANet neck and YOLO head. Adapted from [47].*

For training and fine-tuning purposes YOLOv5 uses the same kind of Mosaic data enhancement as YOLOv4. The Mosaic data enhancement mixes four images into one by random scaling and clipping to enrich the training dataset and to enhance small object learning [48]. Sometimes the clipping results in images that do not show all of the base images. Examples of Mosaic enhanced training images are shown in Figure 10. Training new datasets is further helped with YOLOv5's adaptive anchor box calculation [49]. The adaptive anchor boxes feature is used to calculate suitable bounding box anchors using the training dataset allowing the detector to learn different sized objects more effectively [50]. Additionally, the loss function of YOLOv5 is based on GIOU (generalized intersection over union), which in training helps to preserve the error distance even if predictions and ground truth do not intersect [51].

Figure 10. *YOLOv5's Mosaic enhanced training images created by mixing randomly scaled and clipped base images.*

As premade deployment options YOLOv5 offers multiple pretrained model checkpoints that have been trained with the COCO (Common Objects in Context) dataset. There are 5 different pretrained structures, n, s, m, l and x, that determine the depth of the backbone network and 2 resolution choices, 640x640 and 1280x1280, for each of these structures [52]. A deeper backbone can obtain more features from input images and increase accuracy, but it will also make running the network computationally slower. These pretrained weights can also be used as a basis for fine-tuning and YOLOv5 provides easy interfaces to use them.

YOLOv5 offers a wide variety of different deployment options. It is based on a PyTorch implementation but also provides easy conversions to other formats like TensorFlow, ONNX (Open Neural Network Exchange), CoreML, TensorRT and OpenVINO [52]. This makes it easy to deploy fine-tuned or pretrained YOLOv5 on almost any platform. The shallowest network structures can efficiently be used on mobile devices while the deeper options can be deployed on computation servers to provide simultaneous real-time object detection for multiple video streams.

3.2 Performance Metrics

Many different metrics are used in machine learning to compare the results of various methods and algorithms. These metrics are often based on 4 basic values: true positives, false negatives, false positives and true negatives [53]. In object detection true positives mean detections that correctly localizes and classifies a ground truth object. True negatives, on the other hand, mean areas of an image without objects or detections. False

positives mean falsely localized or classified detections while false negatives mean ground truth objects that were not detected. Example detections depicting these values are shown in Figure 11.

Figure 11. *The upper left image depicts a true positive detection and the upper right images a false positive one. The lower left image shows a false negative as the car in the image is not detected. The lower right is completely true negative as there are no objects or detections.*

These basic values are often used for metrics in object detection, but they are calculated using IoU. Object bounding box prediction is treated as a true positive if the predicted class label is the same as in ground truth and if the predicted bounding box IoU ratio with the ground truth box is higher than a predefined threshold [54]. All bounding boxes that do not fill these criteria are treated as false positives. Ground truth bounding boxes that are not matched by any predictions are regarded as false negatives. The IoU calculation is not used for true negatives as they often are not an interesting result and metrics relying on them are seldom used. Figure 12 demonstrates the IoU calculation's effect on true positives, false positives and false negatives with example detection and ground truth bounding boxes. In the figure green bounding boxes depict ground truth and red bounding boxes depict detections. The figure has 2 images: one on the left and one on the right. With a 0.5 IoU threshold the left image's detection would be counted as a true positive and the right image's detection as a false positive. Additionally, the right image's ground truth object would be counted as false negative as the detection for it was a false positive.

$IoU \approx 0.7$ $IoU \approx 0.15$

Figure 12. *Images with red detection and green ground truth bounding boxes and their IoU scores. The left image depicts a true positive and the right image depicts a false positive and a false negative.*

3.2.1 Precision

Precision is a metric that shows what percentage of model's positive predictions were correct [55]. It can be calculated with the following formula

$$precision = \frac{T_p}{T_p + F_p}$$

where T_p is the number of true positives and F_p is the number of false positives [56]. In object detection the precision score is good at measuring the accuracy of predictions a model is making, but it alone is not enough to make assumptions about the model's true accuracy. False negatives are not accounted for in precision which means that the model can miss almost every object in the input images but still have a precision score of 1.0. Nevertheless, in cases where the most important factor is minimization of false positives, precision can be a very useful metric for evaluating model performance.

3.2.2 Recall

Recall is a metric that indicates the percentage of ground truth that a model has correctly predicted [57]. It is calculated as follows

$$recall = \frac{T_p}{T_p + F_n}$$

where F_n is the number of false negatives [58]. In object detection recall can be used to easily check a model's ability to find all the interesting objects from an input image. But as with precision, recall alone does not give a complete measure of model's accuracy.

The model can achieve a high recall score just by predicting bounding boxes everywhere in an image, which as an output does not convey much information.

3.2.3 Precision-Recall Curve

The precision-recall curve is a plot that links two metrics: precision on the y-axis and recall on the x-axis [59]. The plot can be used to calculate a new metric AUC (area under curve) which represents the area left under the precision-recall curve. This new metric is a combination of precision and recall and can mostly overcome the view limitations of its components. The AUC score considers T_p, F_p and F_n which are the meaningful numbers for object detection. Therefore, it can be used as a complete metric for a model's detection accuracy. However, the AUC score alone does not show the ratio between precision and recall which makes the precision-recall curve also important in a model's accuracy analysis. The AUC also does not consider which IoU threshold is used to calculate T_p, F_p and F_n. An example precision-recall curve with AUC scores created by YOLOv5 framework is show in Figure 13.

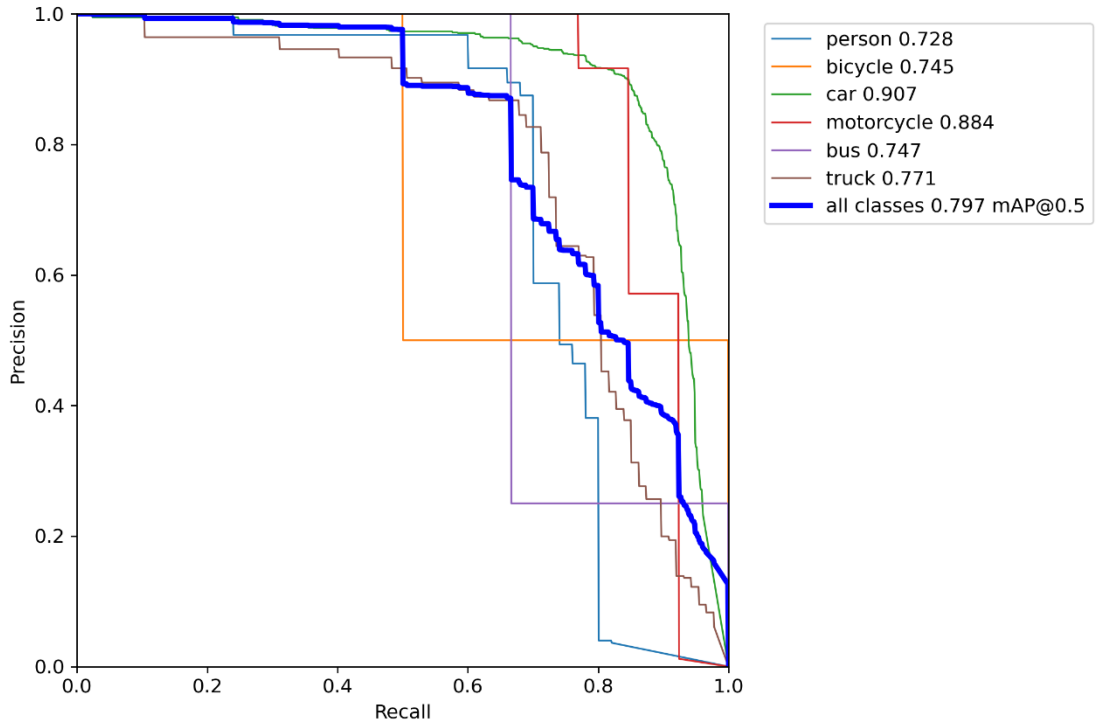

Figure 13. *A precision-recall curve generated by YOLOv5 framework. The numbers in the legend show the AUC scores of the curves representing different classes.*

3.2.4 Mean Average Precision

The mAP (mean average precision) metric is widely used for accuracy and regression performance evaluation in object detection [60]. It is defined as the mean of class specific AP (average precision) scores. AP represents the AUC score of an individual class in a precision-recall curve. In object detection research mAP is often presented with a decimal number added to it in some format like mAP@0.5 [61-64]. The @0.5 notation represents the IoU threshold value used to calculate T_p, F_p and F_n [65]. The threshold value can also be given as a range mAP@.5:.95 which represents the average of mAP scores with threshold values from the given range [66]. The AUC scores in Figure 13 represent AP@0.5 scores for their corresponding classes and all classes score represents mAP@0.5.

With the IoU threshold value mAP offers an informatic performance metric for evaluating object detector accuracy. The metric might still not be optimal for specific tasks as it does not show the exact relation of precision and recall, but generally a higher mAP result with the same IoU threshold is better. In some research AP is used to refer to the same metric as precision and mAP is used to describe the average of the precision metric over all predicted classes [67]. In this thesis AP and mAP are always used to describe the AUC scores of a precision-recall curve.

4. EXPERIMENTS

This chapter goes through experimental setup and the results obtained in the experiments. In section 4.1 the datasets are further reviewed and suitable ones are selected for the experiments. Section 4.2 explains the parameters of the YOLOv5 object detector used in the experiments. Then the algorithmic fine-tuning image selector is proposed in section 4.3. Section 4.4 describes the preliminary experiment used to test the experimental setup and datasets while section 4.5 presents the order and contents of the actual experiments. Section 4.6 shows the experiment results. The final section 4.7 has a summary of the results with observations.

4.1 Datasets

Surveillance and adverse weather object detection datasets were examined and reviewed for use in this thesis' experiments. The examined datasets were chosen from publicly available data for their correspondence with the selected data domain. These datasets were introduced in section 2.3. The datasets were then closely reviewed and two of them were chosen to be used in the experiments: the AAU RainSnow dataset and the DAWN dataset.

Many of the reviewed datasets had some major flaws that made them unsuitable for fine-tuning experiments. The Virat dataset was left out because it only offered bounding boxes and class labels for some objects in its data, which would not be a good basis for fine-tuning. The CADC dataset, on the other hand, had too many objects labelled and many of them were not visible in the images. As there was no easy way to remove the annotations for objects that were not visible, the dataset was also discarded. The MOTChallenge set was also left out as it had many uninteresting objects annotated and did not offer class labels that could be used to sort them out. Also, object class labels were considered crucial for real-life object detection cases and therefore data without object classes could not be accepted.

A more promising dataset was the AU-DETRAC, which provided well labelled data that had surveillance camera view angles. The ignored areas provided by the dataset were insufficient and did not take into account the shaking of the cameras which caused some unlabelled objects to show up in the images. Even with the flaws the data was still usable for fine-tuning but was not chosen for the experiments because of its overall simplicity. The scenes in the data had surveillance view angles, but were quite similar in terms of

weather conditions and surroundings. Additionally, with the ignored areas most of the annotated objects would be in the middle of the image and quite easily recognizable. This of course would be very convenient for computer vision based real-life applications and probably would not need fine-tuning, which made the data less interesting for this research.

The most suitable datasets were chosen to be used in the experiments. The AAU RainSnow dataset was chosen as the main experiment dataset because its data represented both selected data domains and because it could be regarded as data from an actual traffic surveillance application. The dataset contains different weather conditions and has a surveillance view angle. It offers good annotations for all road users, which is good enough for traffic surveillance applications. The set has some flaws in the data, but not too much to prevent its usage. The biggest problem with the data is its size. The dataset has enough data for fine-tuning but not for complete retraining, which removes one possible comparison from the experiments. Nevertheless, it was seen as the best matching public dataset for this research.

The second dataset chosen for the experiments was the DAWN dataset, which contained four different adverse weather condition categories. It had some images with the surveillance view angle mixed with other type of viewpoints. Although the data could not be regarded as real-life application data, it was chosen for its wide variety of different weather conditions that could be present in computer vision solutions. Studying the amount of data needed for adapting to these conditions was seen as an important part of this research. The dataset had similar flaws as the other reviewed sets, but again they were not too prevalent. The number of images in the set was similar to that of AAU RainSnow and would be sufficient for fine-tuning but not enough for complete retraining.

The data in AAU RainSnow was originally split into images from 7 different cameras. This was pruned down to 5 cameras for the experimentation. The annotated images from the cameras were from 2 to 4 different video clips. The images of one clip per camera were chosen as test sets for the corresponding camera while the others were kept as training images. The images were quickly scanned and the ones that had erroneous annotations were removed. The cameras were named with abbreviations of their name in the dataset. The used abbreviations for the cameras were: AAU-Had, AAU-Has, AAU-Hjo, AAU-Ost and AAU-Rin. Figure 14 shows example images from these cameras.

Figure 14. *Example images from AAU RainSnow cameras selected for experiments. The used abbreviation names for the cameras are AAU-Had, AAU-Has, AAU-Hjo, AAU-Ost and AAU-Rin.*

The Dawn dataset was also divided into training and testing images. It did not have specific cameras or video clips but was instead divided by 4 weather condition categories: fog, rain, snow and sand. All these categories held 200 to 300 images, which were evenly split into test and training sets for every category. Table 1 summarizes the numerical features of the 2 selected datasets.

Table 1. *AAU RainSnow and DAWN datasets' numerical features.*

	AAU RainSnow	DAWN
Number of cameras/categories	7	4
Number of images in total	2200	1027
Number of images per camera/category	100-500	200-323
Number of annotated classes	5	6
Average number of annotated instances per image	Around 5	Around 4

4.2 Object Detector

From the group of YOLO based detectors the YOLOv5 object detector was selected to be used in the experiments because of its speed and accuracy combined with a good framework that allows easy usage and versatile deployment options. The selected YOLOv5 network structure was the midway YOLOv5m, as it has adequate depth for feature extraction but is still quite fast to run. The network input resolution was chosen to be 640x640 because images in AAU RainSnow and DAWN were mostly in low resolution. All fine-tuning experiments conducted with the detector used a batch size of 28 and were run for 300 epochs.

4.3 Selection Algorithm for Fine-tuning Images

Fine-tuning experiments were conducted with full training sets and with various number of selected fine-tuning images. The experiments with fewer images than full training sets were conducted with both randomly selected images and images selected algorithmically using the pretrained detection model. The idea in the algorithmic selection was to automatically select good images for fine-tuning training. In applications where object detection is done on camera streams this kind of algorithm could be beneficial as it could automate the process of selecting images for annotation and fine-tuning.

The selection algorithm used the pretrained YOLOv5 object detector to evaluate the images. It selected images based on the number of detected objects and the confidence scores of these detections. The algorithm only uses detections with confidence over 0.15 in the selection. Detections with a confidence score of less than 0.15 were discarded as they were too unreliable to be used in the image selection.

Half of the selected images had the largest number of detected object instances. The number of detected objects was used as the first parameter to ensure that the resulting image batch would have many example instances for feature learning. Large number of example instances would also reduce the risk of overfitting. The second half of the selected images had the lowest average detection confidence. Low detection confidence was a good parameter because low confidence detections show that the pretrained model is not performing well and would require further learning.

The selection algorithm did not take into account the differences in class instance counts which could lead to underrepresentation of some classes in the selected images. Also modifying the selection parameters and the confidence threshold would affect the selection results. This could affect the results of fine-tuning with the selected images but improving and further testing the selection algorithm's parameters and confidence threshold was confined outside of this thesis' scope. A Python class for running this algorithm is presented in appendix A and is also available in Github[1].

4.4 Preliminary Experiment

Before the actual experimentation the datasets and the setup for the experiments were tested with a preliminary experiment. The preliminary experiment was conducted by running individual fine-tuning runs for each of AAU RainSnow's cameras using their respective training sets. Afterwards the results were generated with validation runs of the test

[1] https://github.com/Romeroxx/Object_Detection_Fine-tuning

sets using the pretrained default model and the newly fine-tuned one. The results showed that the AAU RainSnow dataset's person and bicycle classes were difficult to detect. The pretrained model scores were very low and in some cameras the fine-tuning had made improvements, but in others it had decreased the accuracy. The results of the preliminary experiment is presented in appendix B.

The weak performance can mostly be explained with the small size of these objects in the dataset's images and the low amount of object instances. Small objects are harder to detect due to their low resolution and noisiness [68]. The low number of object instances in training data makes fine-tuning harder as there are fewer examples to learn from. This also means that more examples are needed in training and fine-tuning to produce effective results. Additionally, people riding bicycles in the images were only labelled as bicycles and not as a bicycle and a person, unlike in the COCO dataset which is the basis for the pretrained model. Examples of person and bicycle objects in AAU RainSnow dataset are shown in Figure 15.

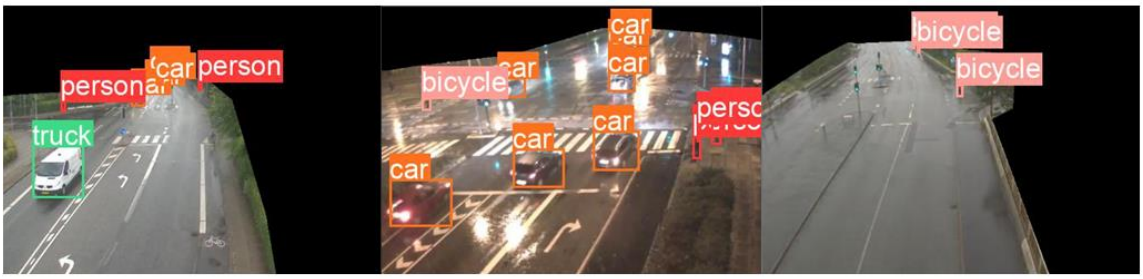

Figure 15. *Example images from AAU RainSnow showing the size of person and bicycle class objects.*

The person and bicycle object classes were decided to be left out of further experiments because of their combination of small size and low object instance count. Using the bicycle and person classes in the experiments would require data with more object instances. The truck and bus classes also had low object instance count in the data, but they were decided to be left in as they were larger in size and would affect the learning of the car class. The DAWN dataset was also checked, and it also had a low object instance count for these classes. For the DAWN dataset bicycles and motorcycles were removed because of their low instance count, but the person class was retained because in DAWN the person object sizes were larger than in AAU RainSnow.

It was also noted during the preliminary experiment that the AAU RainSnow data had several annotations inside the masked areas or annotations for objects that were barely visible. These annotations were mostly on the edges of the images where the masks start or where cars were leaving the image area. These annotations were programmatically removed from the data to reduce the number of annotations inside the masked

areas. Also, the number of marginally visible cars that were about to leave or enter the image area was thus reduced because they were seen as uninteresting. The number of images, remaining classes and their number of object instances for the 5 AAU RainSnow and the 4 DAWN training sets after the modifications done in the preliminary experiment are summarized in Table 2.

Table 2. *The number of images and the number of class instances in AAU RainSnow and DAWN training sets after modification.*

	Number of images	Number of cars	Number of trucks	Number of buses	Number of persons
AAU-Had	89	453	25	0	
AAU-Has	200	823	82	4	
AAU-Hjo	296	1565	195	102	
AAU-Ost	297	1199	107	20	
AAU-Rin	199	385	37	5	
DAWN fog	150	873	97	39	63
DAWN Rain	100	670	109	8	13
DAWN sand	162	936	127	33	84
DAWN snow	102	809	56	12	85

4.5 Conducted Experiments

Fine-tuning experiments were conducted on the selected datasets AAU RainSnow and DAWN with the YOLOv5 object detector. The purpose of the experimentation was to find out how many images are needed for good fine-tuning results with the selected data. Additionally, the proposed data selection algorithm's performance was to be tested. Results from all the experiment runs are presented in categorized tables in appendix B. The results are also more thoroughly analysed in section 4.6.

The experimentation was conducted as 6 separate experiments. Experiment 1 used each of AAU RainSnow's cameras for individual fine-tuning runs. Results were obtained with validation runs on the cameras' test sets. Experiment 2 continued with fine-tuning on the individual cameras with varying number of training images. Fine-tuning runs were made for each camera with 10, 30, 50, 75 and 100 images selected using the selection algorithm and random selection. Experiment 3 used images from 1, 2, 3 or 4 randomly selected AAU RainSnow cameras for fine-tuning while results were gained by validation with a camera that was not in the training set. The last experiment with the AAU RainSnow dataset was Experiment 4. It used 5, 10, 15, 20, and 25 images from every AAU RainSnow camera for fine-tuning with random and algorithmic selection. Validation for these runs was done with all test sets.

After experimenting with AAU RainSnow, the DAWN dataset was used for similar experiments. Experiment 5 used DAWN to fine-tune for individual weather condition categories with 10, 30, 50, 75 and 100 randomly and algorithmically selected images. In Experiment 6 fine-tuning was also run for all weather categories with 5, 10, 15, 20 and 25 images from every category. The resulting models of these runs were tested with the individual categories and with all the categories respectively.

4.6 Results

The results from all 6 Experiments can be found in appendix B under individual experiment headings. The main results were brought up in summarized format in the following sections. In the sections the experiments are gone through in order. The main findings are analysed in the summary chapter.

4.6.1 Experiment 1

The first experiment with the AAU RainSnow data showed the possible accuracy gains of fine-tune training. The results shown in Table 3 present the AP scores for the car class when validation was run with the pretrained model and the fine-tuned one. The results show that fine-tuning with the camera's own image data always improves the accuracy of the predictions.

Table 3. *The first experiment's number of fine-tuning images and AP scores for the car class with the pretrained and the fine-tuned model.*

	AAU-Had	AAU-Has	AAU-Hjo	AAU-Ost	AAU-Rin	Average
Number of fine-tuning images	89	200	296	297	199	216
Fine-tuned car AP	0.895	0.923	0.924	0.902	0.662	0.861
Pretrained car AP	0.798	0.827	0.549	0.703	0.327	0.641

The extent of the improvement seems to vary between the cameras. Some cameras like AAU-Hjo get a large improvement while others like AAU-Had and AAU-Has get a smaller improvement though this difference can probably be attributed to the differences in data. The images of AAU-Had and AAU-Has feature daytime while AAU-Hjo has mostly nighttime scenery in its images. The pretrained model is trained on mostly daytime images so it performs better on the AAU-Had and AAU-Has cameras. The fact that all the cameras except AAU-Rin have around 0.9 AP after fine-tuning shows that the model can learn new features given the data for that, but it also shows the possible learning limit of the

detector. Conclusions about the learning limit of the detector cannot be made with this amount of data as only 100 to 300 images were used for training per camera. The AAU-Rin camera's low scores are probably due to its angle and the glare caused by the wet road surface as is shown in Figure 16.

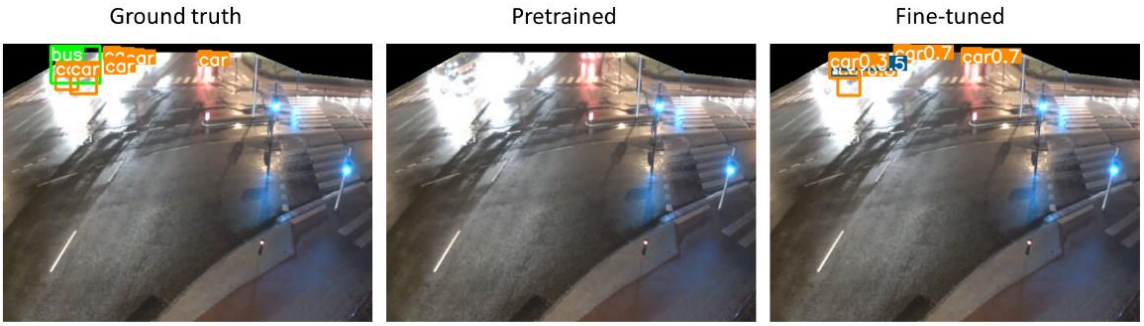

Figure 16. *Images from AAU-Rin camera showing the effects of glare on detection. Images from left show ground truth, pretrained and fine-tuned detections.*

The improvement on bus and truck class AP scores is quite different. On some cameras, such as AAU-Hjo, there is major improvement in the scores but other cameras like AAU-Had have a worse score than the pretrained model. These results can be explained with the class instance counts of the training and test sets. Overall, the AAU RainSnow dataset has a small and varying number of truck and bus class instances while it has about 2 or more car instances per image. Table 4 shows class instance counts and fine-tuned AP scores for these classes on AAU-Had and AAU-Hjo cameras. The other cameras also had similar results.

Table 4. *The first experiment's fine-tuned AP scores and class instance counts for bus and truck classes on AAU-Had and AAU-Hjo cameras.*

	AAU-Had bus	AAU-Had truck	AAU-Hjo bus	AAU-Hjo truck
Class AP	0.082	0.117	0.743	0.439
Training set instances	0	25	102	195
Test set instances	50	49	13	4

From the class instance counts it is easy to see the reason for the fine-tuned model's bad performance with AAU-Had as there are no buses and only a few trucks in AAU-Had's training set. AAU-Hjo on the other hand has a lot more class instances in its training set and is able to produce much better results. Although AAU-Hjo has more training instances for the truck class it produces worse AP score than the bus class. This is due to the small instance count of the test set as the truck class has only 4 test instances which might not represent the class properly. Another contributing factor is the variety of

the truck and bus classes. Although these objects are often large in size, their appearance varies and a model would need examples of many variants to effectively learn their features.

4.6.2 Experiment 2

Fine-tuning with varying number of images in Experiment 2 showed that boosting the performance of the model does not require all the training images. The runs with varying number of images were repeated 5 times for AAU-Has and AAU-Hjo cameras to produce statistically more relevant results. The results of these runs are presented in Figure 17 and Figure 18 where the charts' bars are showing the average car class AP score over the 5 separate runs. The error bars also show the minimum and maximum AP scores produced in the experiment runs.

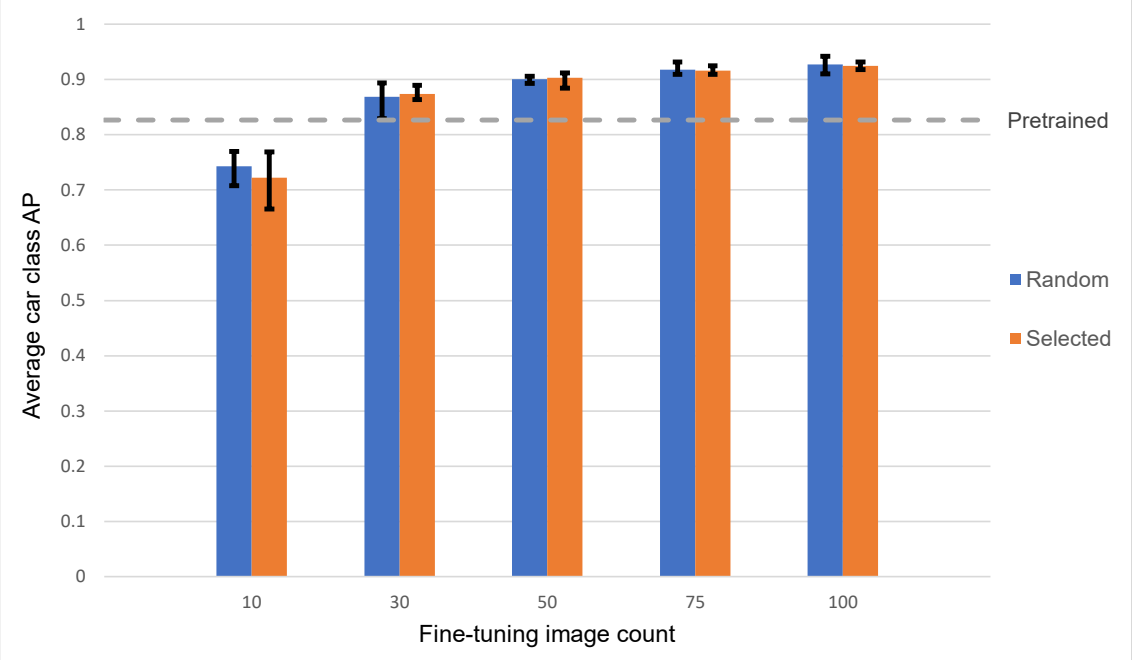

Figure 17. *Experiment 2 average car class AP scores over 5 fine-tuning runs with varying number of training images from the AAU-Has camera training set.*

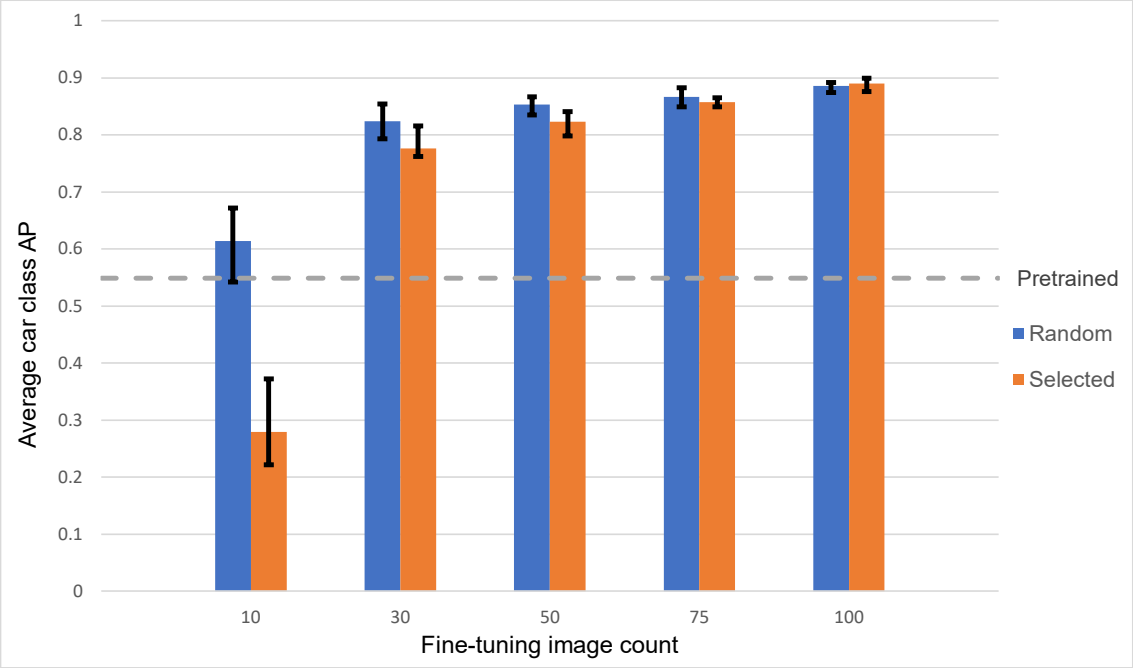

Figure 18. *Experiment 2 average car class AP scores over 5 fine-tuning runs with varying number of training images from the AAU-Hjo camera training set.*

Both figures show that fine-tuning with 30 images is enough to produce a better AP score than what the pretrained model could provide. It can also be seen that the accuracy improves more significantly from 10 to 50 images than from 50 to 100. The AAU-Has model achieves its best score at 75 images and seems to saturate after that while AAU-Hjo is still further improved by fine-tuning with more than 100 images as can be seen from comparing the fine-tuning run with 100 images to the results for AAU-Hjo in Experiment 1. The AP score improvements could be seen on the actual detection level, as is shown in Figure 19, which presents an image from AAU-Has and AAU-Hjo test sets with ground truth, pretrained and fine-tuned detections. The AAU-Has camera model used for the Figure 19 was fine-tuned with 50 images and the AAU-Hjo camera model with 75.

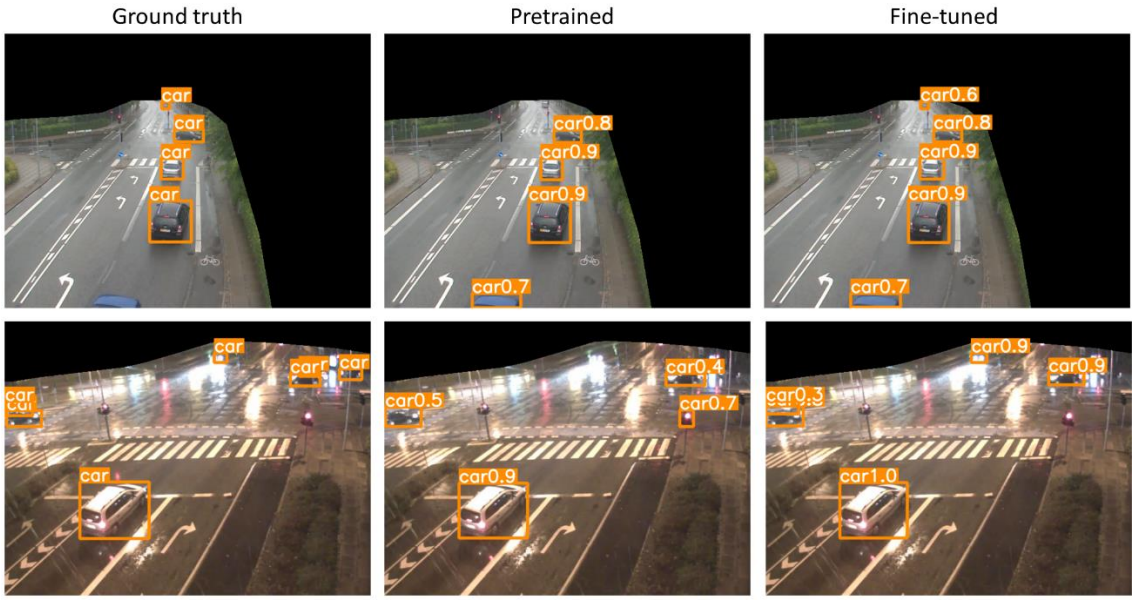

Figure 19. *Images from AAU RainSnow AAU-Has and AAU-Hjo test sets with ground truth, pretrained and fine-tuned detections. The AAU-Has model fine-tuning used 50 images while the AAU-Hjo model used 75.*

In AAU-Has the differences between runs with randomly and algorithmically selected images seem to be very small. Both selection practices seem to produce very even results which suggests that all images in AAU-Has training set are of good quality and equally effective as fine-tuning images. AAU-Hjo on the other hand produces better results with randomly selected data when the fine-tuning image count is low. This is probably due to the fact that the AAU-Hjo camera has mostly night-time images, but the selection algorithm uses the pretrained model which makes better detections from daytime images. At a low fine-tuning image count this could lead the selection algorithm to prefer the few daytime images preventing the model from learning night-time features properly. This hypothesis is affirmed by the selection algorithm's good performance with higher image quantities.

The other cameras produced similar results as AAU-Has and AAU-Hjo except for AAU-Rin which got a larger improvement by having more than 100 training images. Also, the selection algorithm seemed to perform better with low image quantities than with higher ones. This was again contributed to the glare effect which completely hides some of the objects in AAU-Rin's images. The selection algorithm probably does not detect anything in the glare and prioritises images without it, which helps to a certain extent but has its limit. If images with the glare effect are not selected for the training set, then the model cannot learn the objects distorted by it.

4.6.3 Experiment 3

The Experiment 3 tested the generalization ability of the fine-tuned models inside the AAU RainSnow dataset. The results of Experiment 3 showed that the similarity of the training images and test images had a larger influence on the results than having more training cameras. If the first camera used as a training set was similar to the test set, then the resulting scores would be almost as high as with 4 training cameras and a larger training set. Still, having more cameras usually resulted in better results as is seen from Table 5 and Table 6, which show results for AAU-Had and AAU-Ost cameras. The training set cameras were selected randomly and the results of the random selections are shown in the tables.

Table 5. *Experiment 3 fine-tuned car class AP scores for AAU-Had camera.*

Training sets	AAU-Has	AAU-Has, AAU-Rin	AAU-Has, AAU-Rin, AAU-Hjo	AAU-Has, AAU-Rin, AAU-Hjo, AAU-Ost
AAU-Had car AP	0.875	0.875	0.907	0.904

Table 6. *Experiment 3 fine-tuned car class AP scores for AAU-Ost camera.*

Training sets	AAU-Rin	AAU-Had, AAU-Hjo	AAU-Had, AAU-Hjo, AAU-Rin	AAU-Had, AAU-Hjo, AAU-Rin, AAU-Has
AAU-Ost car AP	0.495	0.783	0.795	0.836

The results for AAU-Had show that fine-tuning with just 1 camera resulted in the same level of accuracy as training with 4 cameras. On the other hand, testing AAU-Ost with a model fine-tuned with 4 cameras resulted in a much higher score than fine-tuning with 1 camera. Both test sets also show slightly stagnant states, where adding training images from a new camera has little effect on the results. This shows that the similarity of the fine-tuning images and the test images is more important than the quantity of the training images.

4.6.4 Experiment 4

Experiment 4 tested fine-tuning for all 5 AAU RainSnow cameras (AAU-Had, AAU-Has, AAU-Hjo, AAU-Ost and AAU-Rin) at the same time. The test was repeated 5 times as in Experiment 2 to further validate the results. Figure 20 shows the resulting car class AP scores over the 5 runs with error bars showing minimum and maximum results.

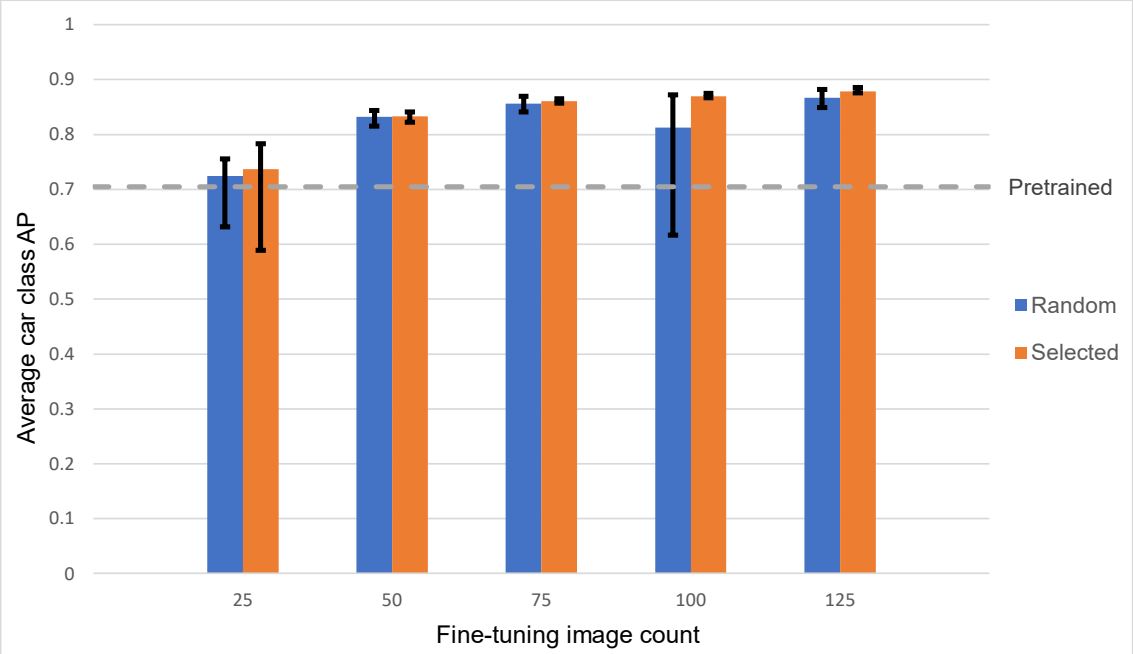

Figure 20. *Experiment 4 AP scores for the car class when fine-tuned with varying number of images from all 5 of the AAU RainSnow cameras. The scores are an average over 5 runs where the error bars show the minimum and maximum.*

The results indicate that fine-tuning with 5 images from every camera is enough to produce a model that performs better overall than the pretrained one. The scores are improved by adding more images but seem to somewhat saturate after using more than 75 images in total for fine-tuning. The AP score at the saturation point is very close to the average of the individual camera scores obtained in Experiment 1. Figure 21 shows the effects of fine-tuning with 15 images from all 5 cameras with the AAU-Had test set.

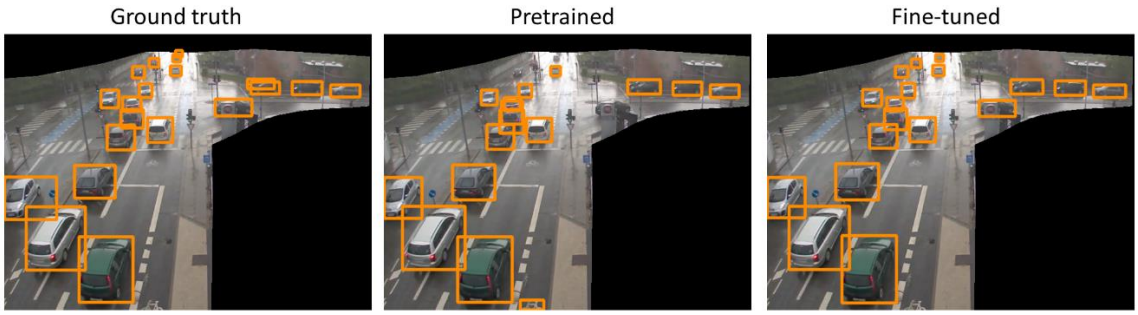

Figure 21. *An image from AAU RainSnow Had camera showing ground truth on the left and pretrained detections in the middle. The rightmost image's detections are from a model that was fine-tuned with 15 images from every AAU RainSnow camera.*

The results show that randomly and algorithmically selected fine-tuning images produce similar results. The selection algorithm seems to get slightly better results, but this is mostly just a marginal difference. The biggest difference is in the run with 100 fine-tuning images where one of the 5 runs with random images has resulted in a noticeably lower

score. This shift could be caused to the random selection of the fine-tuning images, but in that case, there should be more deviation in the other tests as well. For that reason, it is more likely that the stochasticity of the DNN training process caused it to end up with less optimal weights on that single run. Nevertheless, the minimum and maximum scores for the runs would suggest that the algorithmic selection produces less deviation in the results, although the differences are not major. An exception to this is the run with 25 images where both selection criteria have a large deviation in the results. The deviation in those runs is caused by the low number of training images. With a small training set it is more likely that the model does not learn all the important features. A small learning set also increases the possibility of overfitting.

4.6.5 Experiment 5

The DAWN dataset was tested in Experiment 5 with individual weather category runs. The results on these categories were similar to results in Experiment 1. Fine-tuning with 30 images in all categories was enough to produce a better score than what the pre-trained model could achieve. Almost top-level accuracy was achieved with 50 or 75 images. Figure 22 shows the average AP scores for 5 runs on every image count with the rain weather category. One of the runs with 75 fine-tuning images produced almost 0 AP scores with all classes. As no other run behaved similarly this result was dismissed as a by-product of stochasticity in the DNN training process.

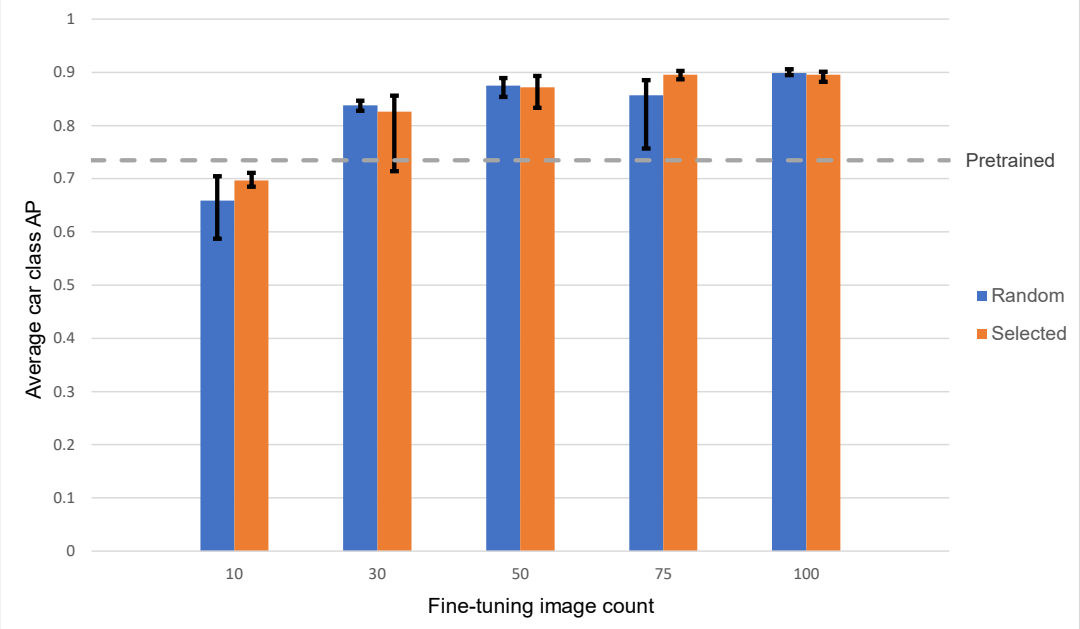

Figure 22. *Average car class AP scores over 5 runs for DAWN dataset's rain category in Experiment 5. The error bars show minimum and maximum from the 5 runs.*

Results in Figure 22 show that randomly and algorithmically selected images produce scores of the same level. Both have some deviation in the results with lower number of images but get more stable with larger training image batches. The runs with fog, snow and sand weather categories produced slightly better results with the selection algorithm, but again these differences were mostly marginal. Figure 23 shows the detection accuracy improvements produced by the 50-image fine-tuning with the rain category.

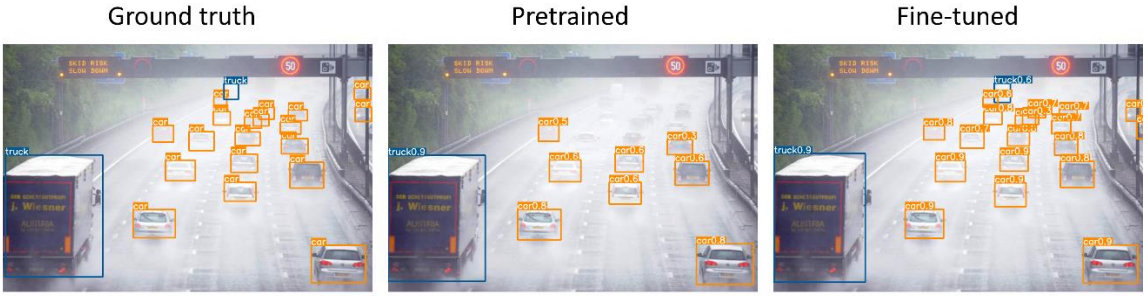

Figure 23. *Ground truth, pretrained and 50-image fine-tuned detections on DAWN dataset's rain category.*

Similar to the AAU RainSnow the DAWN dataset also has a varying number of object instances in other classes than the car class which has around 4 instances per image. The AP scores for the truck class were similar to the scores in Experiment 1. Having more truck instances in the training set improved the resulting accuracy and with sufficient number of instances the resulting model could surpass the pretrained model. The AP scores for the bus class were similar to the truck class as the scores increased with more training samples but they would not surpass the pretrained model in every weather category. Additionally, none of the fine-tuned models were able to surpass the pretrained model's accuracy with the person class. This shows that more training instances are needed when fine-tuning for smaller sized classes. Some categories like rain had an unexpectedly good score for the person class considering the number of training instances which is mostly due to highly optimized pretrained weights. Table 7 shows pretrained scores and fine-tuned score averages over 5 runs for person, bus and truck classes in the rain category.

Table 7. *Fine-tuned and pretrained person, bus and truck class AP scores for the DAWN dataset's rain category. Fine-tuned scores are averages over 5 runs.*

	Person	Bus	Truck
Fine-tuned AP	0.612	0.373	0.680
Pretrained AP	0.735	0.221	0.457
Instance count in training set	13	8	109

4.6.6 Experiment 6

All of DAWN dataset's categories (fog, rain, sand and snow) were tested together in Experiment 6. For this final experiment the runs were again repeated 5 times for statistical relevance. The resulting scores are compiled into Figure 24 which shows the average car class AP scores over the 5 runs with minimums and maximums.

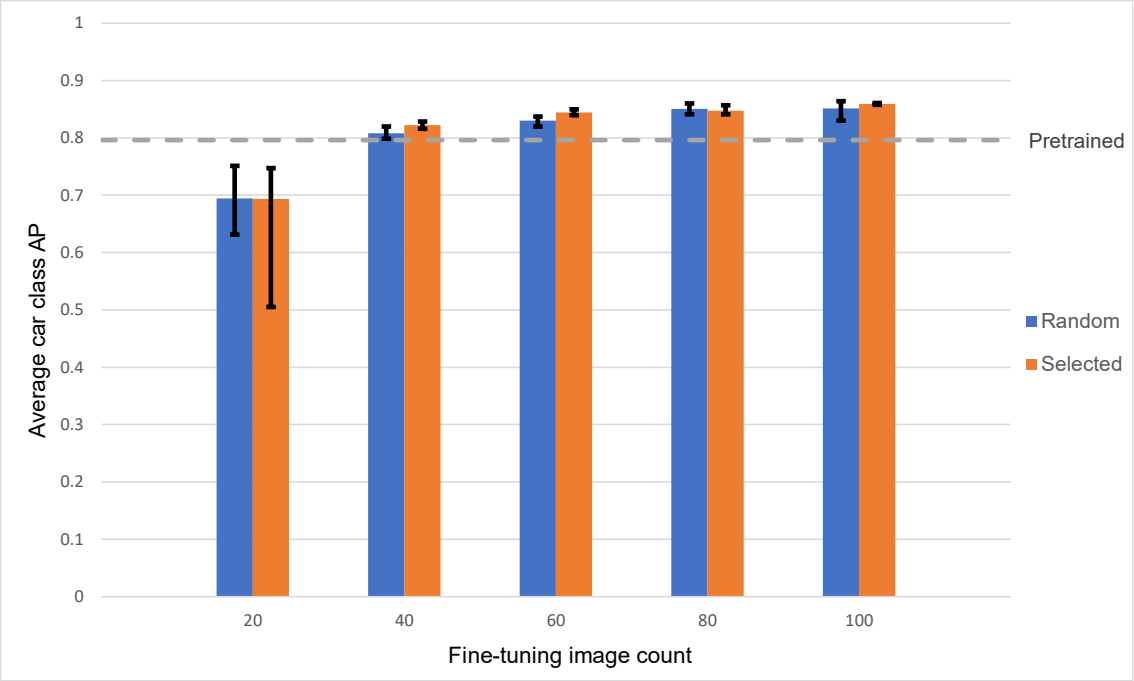

Figure 24. *DAWN all categories fine-tuning results with varying number of training images and the pretrained result from Experiment 6 featuring AP scores for the car class. The scores are an average over 5 runs where the error bars show the minimum and maximum.*

The scores are similar to Experiment 4, but the results are not identical. The fine-tuned results achieve a higher score than the pretrained one with 40 images in total while in Experiment 4 25 images were enough. The accuracy gain seems to stop after 60 images which is in line with earlier results, but the improvement compared to the pretrained model is lower than with AAU RainSnow dataset. This difference is mostly caused by differences in the data as the pretrained model had a lot of errors with some cameras in AAU RainSnow, but performed quite well with the DAWN data. The selection algorithm produced slightly better results than random selection and had a bit less deviation in its results with higher number of training images, which is similar to Experiment 4. Figure 25 shows example detections from the pretrained model and from the model fine-tuned with 15 images from all categories.

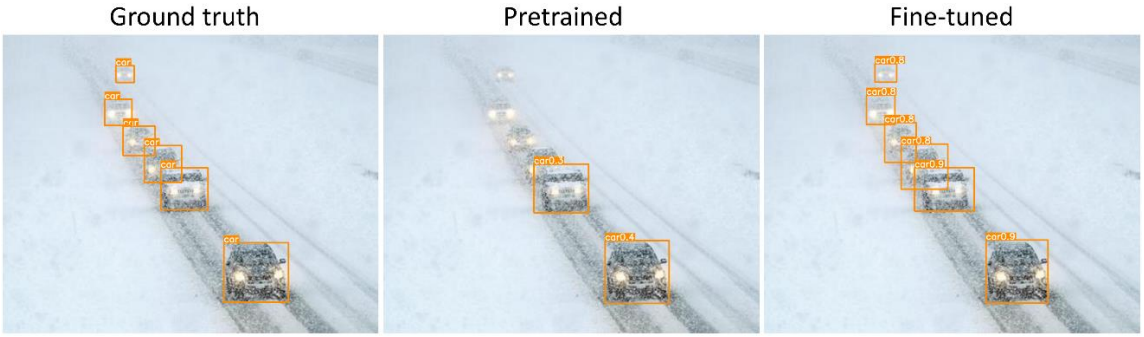

Figure 25. *Ground truth, pretrained and fine-tuned detections on an image from DAWN dataset's snow category. The fine-tuning was done with 15 images from every DAWN weather category.*

4.7 Summary

The conducted experiments show that fine-tuning can be used to improve the detection accuracy of the YOLOv5 detector. This result was obtained with two dataset both of which produced similar results. In the experiments it was noticed that there is a limit to how much the detectors accuracy can be improved by fine-tuning. This limit was reasoned to be caused by limitations in the detector architecture and the challenges in the data. With real data there are always cases where objects are too obstructed or otherwise too ambiguous for detection. In most of the experiments the limit was already reached before using all the data in fine-tune training. There is no perfect object detector, so a limit has to exist for each detector. Additionally, both used datasets had errors in the ground truth which also lowers the maximum achievable score.

The results show that 50 or 75 fine-tuning images from one camera or weather category can be enough to reach the detector's accuracy limit. The same accuracy limit can be reached with around 15 images per camera or weather category when fine-tuning the detector for multiple cameras or categories. Adding more images in both cases can further improve the accuracy, but the gains are clearly smaller. Also, having less than 30 images for fine-tuning in total is shown to result in large deviation between models trained with the same data. Having too few fine-tuning images can result in a model that is worse than the original pretrained model. These results would mean that the optimal number of fine-tuning images is around 70 for 1 camera or 15 per camera for several cameras. However, the experiments conducted in this thesis only covered fine-tuning with a small number of images and complete retraining with thousands of images was not experimented.

The results also showed that successful fine-tuning with a small number of images requires sufficient number of object instances in the training data. When fine-tuning with a

small data batch, such as 50 images, the detection accuracy for classes with a low number of instances in training data was significantly lower compared to classes with high number of instances. This is caused by the model overlearning the features of the scarce example instances resulting in a lowered score with the testing data. It was seen that for car sized objects an average of 2 or more instances per image was enough to get good results. For larger objects like trucks even less could be enough, but more instances always produced a better overall score. However, the same number of instances was not sufficient for small objects like persons which shows that learning smaller sized objects requires more example class instances.

Automatic fine-tuning image selection was tested in the experiments. It was shown that the proposed selection algorithm with the pretrained model could slightly improve fine-tuning results when compared to random image selection. This improvement was not present in all test cases but Experiments 4 and 6 showed a slight improvement in accuracy with the selection algorithm. Additionally, with larger number of fine-tuning images the selection algorithm seemed to reduce the deviation of the results when compared to random selection. These results are not enough to statistically prove that the algorithmic selection produces better results than random image selection. Nevertheless, it is shown that the proposed selection method does not lower the results, so using it would not impair fine-tuning.

The algorithmic selection could also be used to collect data from real life camera streams. The selection algorithm could be set do selection on a camera’s live video stream to automatically collect fine-tuning images. A benefit of the algorithmic selection is that it could be modified to suit the needs of the specific case. The algorithm could be modified to only collect images with some rare class or to focus on small or big detections. However, the downside of algorithmic selection is that it does not work as well on data that is unevenly challenging to the pretrained model. The algorithm could select mostly simple images and discard many images that would be beneficial for learning, as was seen in Experiment 2 with the AAU-Hjo camera. Though this problem can be alleviated by modifying the algorithm to prioritise low confidence images or by changing to a different detector.

5. CONCLUSIONS

This thesis studied the number of images that are needed to produce improved fine-tuning results in object detection. The research was motivated by the need for precise object detectors in various real-life applications where pretrained models are not good enough. The goal was to find out an optimal number of fine-tuning images to use, as labelling images can be time-consuming and costly if crowdsourcing is not possible. To reach this goal, fine-tuning experiments were conducted on publicly available data using the YOLOv5 object detector. Data was chosen to represent surveillance camera view angles or adverse conditions, both of which could be present in a real scenario. In the experiments fine-tuning image constraints were also surveyed and an algorithmic image selector was proposed to further facilitate fine-tuning.

For the experiments, publicly available datasets were examined and reviewed. Two of the reviewed sets were used in the experimentation: traffic surveillance dataset AAU RainSnow and various weather conditions dataset DAWN. The experiments on the datasets tested overall fine-tuning benefits on single traffic cameras and weather categories with different image quantities. Also, experiments were made to fine-tune a model for several cameras or weather categories. The results showed that there is a limit on how much fine-tuning can improve the accuracy and 50 to 75 images were enough to produce results that are close to this limit for a single camera or category. Adding more images would only marginally improve the results and having less than 30 images would result in overlearning and worse accuracy than with the pretrained model. Similar results were obtained by fine-tuning with 15 images per camera or weather category when using multiple sources.

The experimental results also showed that to produce good results with only 75 images, the data needs to contain a sufficient number of object instances. Good results were achieved with car-sized objects when there were 2 or more instances on average. Larger objects could be learned with fewer instances but having more would improve the accuracy. It was also noted that smaller objects require notably more example instances than larger objects.

Algorithmic fine-tuning image selection was also tested. The proposed selector used the pretrained model to make predictions and selected images with high object count or low average confidence. This algorithm was tested against random selection and the exper-

iments showed that the algorithmic selection can improve the results slightly and decrease deviation between models fine-tuned with the same dataset. However, the improvement was mostly marginal and was not the case in all experiments, as some conditions in the data impaired the selector's performance. Overall, the selector produced results that were at least as good as with random selection though, and so can be helpful in gathering fine-tuning images from a real camera stream, in which it is hard to say when random selection should be done.

From the experimental results it was concluded that fine-tuning a pretrained object detector with a small number of images is a viable option in scenarios where improved accuracy is required, but data sensitivity prevents crowdsourcing. The needed number of images for improved results is low enough to be carried out as self-annotation. This need is further reduced if there are multiple similar image sources, and a single model can be fine-tuned with a small number of images from all of them. The self-annotation process can be expedited with algorithmic image selection which can lower the time spent on selecting good images for annotation.

All of the results in this thesis' experimentation point to a similar number of required fine-tuning images for cost optimal accuracy improvement. This makes the result credible, but the experimentation was done with only two datasets and one object detector. Also because of public data limitations the experiments could provide reliable results only on the car class as other classes were underrepresented in the chosen datasets. To make the results statistically more relevant, further research with different datasets and other object classes is required.

Another interesting future research topic is to find out the how many images is required to learn a new classes with fine-tuning. New classes that are similar to the ones known by the pretrained model could be learned through fine-tuning. Information on requirements for learning a new class could be very useful, as real applications often need detections from classes other than pretrained ones. Future work could also include further testing with the image selection algorithm. Experiments conducted in this thesis only used one type of configuration for the algorithm and already found out that it could be beneficial. Additional testing with different configuration and focus points could yield better results and so could further improve the cost efficiency of fine-tuning with self-annotated data.

REFERENCES

[1] P. Chen, Y. Shi, Q. Zheng, Q. Wu, State-of-the-art of Object Detection Model Based on YOLO. 2020 International Conference on Computer Network, Electronic and Automation (ICCNEA), 2020, pp. 101–105.

[2] E. Arkin, N. Yadikar, Y. Muhtar, K. Ubul, A Survey of Object Detection Based on CNN and Transformer, 2021 IEEE 2nd International Conference on Pattern Recognition and Machine Learning (PRML), 2021, pp. 99–108.

[3] H. Vaidwan, N. Seth, AS. Parihar, K. Singh, A study on transformer-based Object Detection. 2021 International Conference on Intelligent Technologies (CONIT), 2021, pp. 1–6.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, 2021 International Conference on Learning Representations (ICLR), 2021.

[5] Model Zoo, PyTorch, Available (referenced 04.03.2022): https://pytorch.org/serve/model_zoo.html

[6] TensorFlow 2 Detection Model Zoo, GitHub, Available (referenced 04.03.2022): https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

[7] I. Elezi, Z. Yu, A. Anandkumar, L. Leal-Taixe, JM. Alvarez, Not All Labels Are Equal: Rationalizing The Labeling Costs for Training Object Detection, ArXiv.Org, 2021, Available: https://arxiv.org/abs/2106.11921

[8] M. Xu, Y. Bai, B. Ghanem, Missing Labels in Object Detection, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2019, pp. 1-10.

[9] H. Su, J. Deng, L. Fei-Fei, Crowdsourcing annotations for visual object detection, AAAI Workshop - Technical Report, 2012, pp. 40–46.

[10] X. Zhang, F. Yan, Y. Zhuang, H. Hu, C. Bu, Using an Ensemble of Incrementally Fine-Tuned CNNs for Cross-Domain Object Category Recognition, IEEE access, 2019, Vol 7, pp. 33822–33833.

[11] A. Zhang, ZC. Lipton, M. Li, AJ. Smola, Dive into Deep Learning, ArXiv.Org, 2021, Available: https://arxiv.org/abs/2106.11342

[12] S. Arif, F. Shafait, Table Detection in Document Images using Foreground and Background Features, 2018 Digital Image Computing: Techniques and Applications (DICTA), 2018, pp. 1–8.

[13] JU. Kim, Y. Man Ro, Attentive Layer Separation for Object Classification and Object Localization in Object Detection, 2019 IEEE International Conference on Image Processing (ICIP), 2019, pp. 3995–3999.

[14] J. Fan, J. Lee, I. Jung, Y. Lee, Improvement of Object Detection Based on Faster R-CNN and YOLO, 2021 36th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), 2021, pp. 1–4.

[15] C-C. Wang, H. Samani, C-Y. Yang, Object Detection with Deep Learning for Underwater Environment, 2019 4th International Conference on Information Technology Research (ICITR), 2019, pp. 1–6.

[16] Z. Zou, Z. Shi, Y. Guo, J. Ye, Object Detection in 20 Years: A Survey, ArXiv.Org, 2019, Available: https://arxiv.org/pdf/1905.05055.pdf

[17] X. Long, Z. Zheng, Y. Chi, R. Liu, A Mixed Two-stage Object Detector for Image Processing of Power System Applications, 2020 IEEE 20th International Conference on Communication Technology (ICCT), 2020, pp. 1352–1355.

[18] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587.

[19] K. He, X. Zhang, S. Ren, J. Sun, Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, IEEE transactions on pattern analysis and machine intelligence, 2015, Vol. 37, Iss. 9, pp. 1904–1916.

[20] R. Girshick, Fast R-CNN, 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440–1448.

[21] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal network, Advances in Neural Information Processing Systems, 2015, pp. 91–99.

[22] W. Hongtao, Y. Xi, Object Detection Method Based On Improved One-Stage Detector, 2020 5th International Conference on Smart Grid and Electrical Automation (ICSGEA), 2020, pp. 209–212.

[23] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C-Y. Fu, A. Berg, SSD: Single Shot MultiBox Detector, Computer Vision – ECCV, 2016, pp. 21–37.

[25] T-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar, Focal Loss for Dense Object Detection, IEEE transactions on pattern analysis and machine intelligence, 2020, Vol. 42, Iss. 2, pp. 318–327.

[26] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, Q. Tian, CenterNet: Keypoint Triplets for Object Detection, 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 6568–6577.

[27] J. Bi, Z. Zhu, Q. Meng, Transformer in Computer Vision, 2021 IEEE International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI), 2021, pp. 178–188.

[28] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, End-to-End Object Detection with Transformers, Computer Vision – ECCV 2020, Cham: Springer International Publishing, 2020, pp. 213–229.

[29] X. Dai, Y. Chen, J. Yang, P. Zhang, L. Yuan, L. Zhang, Dynamic DETR: End-to-End Object Detection with Dynamic Attention, 2021 IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 2968–2977.

[30] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, et al. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, 2021 IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 9992–10002.

[31] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, et al. DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection, ArXiv.Org, 2022, Available: https://arxiv.org/pdf/2203.03605v1.pdf

[32] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, et al. Swin Transformer V2: Scaling Up Capacity and Resolution, ArXiv.Org, 2021, Available: https://arxiv.org/pdf/2111.09883v1.pdf

[33] Y. Wei, S. Liu, J. Sun, L. Cui, L. Pan, L. Wu, Big Datasets for Research: A Survey on Flagship Conferences, 2016 IEEE International Congress on Big Data (BigData Congress), 2016, pp. 394–401.

[34] The VIRAT Video Dataset, Kitware, Available (referenced 08.03.2022): https://viratdata.org/

[35] UA-DETRAC Benchmark Suite, University at Albany, Available (referenced 07.03.2022): https://detrac-db.rit.albany.edu/

[36] Canadian Adverse Driving Conditions Dataset, University of Waterloo, Available (referenced 09.03.2022): http://cadcd.uwaterloo.ca/

[37] DAWN, Mendeley Data, Available (referenced 09.03.2022): https://data. mendeley.com/datasets/766ygrbt8y/3

[38] H. Hedayati, BJ. McGuinness, MJ. Cree, JA. Perrone, Generalization Approach for CNN-based Object Detection in Unconstrained Outdoor Environments. International Conference Image and Vision Computing New Zealand, 2019.

[39] AAU RainSnow Traffic Surveillance Dataset, Kaggle, Available (referenced 10.03.2022): https://www.kaggle.com/aalborguniversity/aau-rainsnow

[40] Multiple Object Tracking Benchmark, MOTChallenge, Available (referenced 11.03.2022): https://motchallenge.net/

[41] G. Chen, X. Bai, G. Wang, L. Wang, X. Luo, M. Ji, et al. Subsurface Voids Detection from Limited Ground Penetrating Radar Data Using Generative Adversarial Network and YOLOV5, The Institute of Electrical and Electronics Engineers, Inc (IEEE) Conference Proceedings, 2021.

[42] Z. Feng, L. Guo, D. Huang, R. Li, Electrical Insulator Defects Detection Method Based on YOLOv5, Proceedings of 2021 IEEE 10th Data Driven Control and Learning Systems Conference, DDCLS, 2021, pp. 979–984.

[43] AK. Shetty, I. Saha, RM. Sanghvi, SA. Save, YJ. Patel, A Review: Object Detection Models, 2021 6th International Conference for Convergence in Technology (I2CT), 2021, pp. 1–8.

[44] Y. Kun, H. Man, L. Yanling, Multi-target Detection in Airport Scene Based on Yolov5, 2021 IEEE 3rd International Conference on Civil Aviation Safety and Information Technology (ICCASIT), 2021, pp. 1175–7.

[45] U. Nepal, H. Eslamiat, Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs, Sensors (Basel, Switzerland), 2022, Vol. 22, Iss. 2

[46] E. Cengil, A. Cinar, M. Yildirim, A Case Study: Cat-Dog Face Detector Based on YOLOv5, 2021 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), 2021, pp. 149–153.

[47] R. Xu, H. Lin, K. Lu, L. Cao, Y. Liu, A forest fire detection system based on ensemble learning, MDPI Forests, 2021, Vol. 12, Iss. 2, pp. 1–17.

[48] H. Wang, S. Sun, X. Wu, L. Li, H. Zhang, M. Li, et al. A YOLOv5 Baseline for Underwater Object Detection, OCEANS 2021: San Diego – Porto, 2021, pp. 1–4.

[49] X. Song, W. Gu, Multi-objective real-time vehicle detection method based on yolov5, Proceedings - 2021 International Symposium on Artificial Intelligence and its Application on Media, ISAIAM 2021, 2021, pp. 142–145.

[50] S. Li, Y. Li, Y. Li, M. Li, X. Xu, YOLO-FIRI: Improved YOLOv5 for Infrared Image Object Detection, IEEE access, 2021, Vol. 9, 141861–141875.

[51] J. Tang, S. Liu, B. Zheng, J. Zhang, B. Wang, M. Yang, Smoking Behavior Detection Based on Improved YOLOv5s Algorithm, 2021 9th International Symposium on Next Generation Electronics, ISNE 2021, 2021.

[52] G. Jocher, Yolov5, GitHub, Available (referenced 15.03.2022): https://github.com/ultralytics/yolov5

[53] D. Julian, S. Raschka, J. Hearty, Python: Deeper Insights into Machine Learning, Module 1, Chapter 6, Looking at different performance evaluation metrics, Packt Publishing, 2016.

[54] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, et al. Deep Learning for Generic Object Detection: A Survey, International journal of computer vision, 2019, Vol. 128, Iss. 2, pp. 261–318.

[55] SA. Khan, Z. Ali Rana, Evaluating Performance of Software Defect Prediction Models Using Area Under Precision-Recall Curve (AUC-PR), 2019 2nd International Conference on Advancements in Computational Sciences (ICACS), The University of Lahore, 2019, pp. 1–6.

[56] I. Cherepanov, A. Mikhailov, A. Shigarov, V. Paramonov, On automated workflow for fine-tuning deepneural network models for table detection in document images, 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), Croatian Society MIPRO, 2020, pp. 1130–1133.

[57] A. Bradley, RP. Duin, P. Paclik, TC. Landgrebe, Precision-recall operating characteristic (P-ROC) curves in imprecise environments, 18th International Conference on Pattern Recognition (ICPR'06), 2006, pp. 123–127.

[58] MR. Prabowo, N. Hudayani, S. Purwiyanti, SR. Sulistiyanti, FXA. Setyawan, A moving objects detection in underwater video using subtraction of the background model, 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), 2017, pp. 1–4.

[59] J. Keilwagen, I. Grosse, J. Grau, Area under precision-recall curves for weighted and unweighted data, PloS one, 2014, Vol. 9, Iss. 3.

[60] H. Zhu, H. Wei, B. Li, X. Yuan, N. Kehtarnavaz, A review of video object detection: Datasets, metrics and methods, Applied sciences, 2020, Vol. 10, Iss. 21, pp. 1–24

[61] A. Tejada-Mesias, I. Dongo, Y. Cardinale, J. Diaz-Amado, ODROM: Object Detection and Recognition supported by Ontologies and applied to Museums, 2021 XLVII Latin American Computing Conference (CLEI), 2021, pp. 1–10.

[62] M-C. Le, M-H. Le, Human Detection and Tracking for Autonomous Human-following Quadcopter, Proceedings of 2019 International Conference on System Science and Engineering, ICSSE, 2019, pp. 6–11.

[63] J. Huang, M. Yang, H. Zhi, L. Xiang, H. Zhang, Y. Wu, Small Object Detection of Non-standard THT Solder Joints Based on Improved YOLOv3, 2021 17th International Conference on Computational Intelligence and Security (CIS), 2021, pp. 608–611.

[64] T, Zhang, L. Li, An Improved Object Detection Algorithm Based on M2Det, 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), 2020, pp. 582–585

[65] D. Patel, F. Patel, S. Patel, N. Patel, D. Shah, V. Patel, Garbage Detection using Advanced Object Detection Techniques, 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 2021, pp. 526–531.

[66] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, IEEE transactions on pattern analysis and machine intelligence, 2017, Vol. 39, Iss. 6, pp. 1137–1149.

[67] F. Joy, V. Vijayakumar, Multiple object detection in surveillance video with domain adaptive incremental fast rcnn algorithm, Indian journal of computer science and engineering, 2021, Vol. 12, Iss. 4, pp. 1018–1026.

[68] P. Fang, Y. Shi, Small Object Detection Using Context Information Fusion in Faster R-CNN, 2018 IEEE 4th International Conference on Computer and Communications (ICCC), 2018, pp. 1537–1540.

APPENDIX A: SELECTION ALGORITHM PYTHON CLASS

```
import torch
class Selector:

    def __init__(self, selected_labels, selection_size,
                 label_multipliers=[]):
        """
        PARAMS:
            selected_labels: List of COCO label strings representing
                             the object classes that will be counted
                             for the selection.
            selection_size: Number of images to select
            label_multipliers: List of multipliers for object classes
        """

        # Initialize YOLOv5 detection model
        self.model = torch.hub.load('ultralytics/yolov5', 'yolov5m')
        self.model.conf = 0.15
        self.model.iou = 0.45

        self.selected_labels = selected_labels
        self.reset(selection_size)


    def reset(self, selection_size, label_multipliers=[]):
        """
        PARAMS:
            selection_size: Number of images to select
            label_multipliers: List of multipliers for rare classes
        """

        count_selection_size = int(selection_size/2)
        if selection_size % 2:
            count_selection_size += 1

        # Reset lists for image infromation saving
        self.selected_image_counts = [0] * count_selection_size
        self.count_image_scores = [2.0] * count_selection_size
        self.count_selected_images = [None] * count_selection_size
        self.lowest_count = 0
        self.lowest_count_index = 0

        self.selected_image_scores = [2.0] * int(selection_size/2)
        self.score_selected_images = [None] * int(selection_size/2)
        self.highest_score = 2.0
        self.highest_score_index = 0

        # Generate dummy label multipliers if not given
        if not label_multipliers:
            label_multipliers = [1] * len(self.selected_labels)

        self.multipliers = {}
        for i, label in enumerate(self.selected_labels):
            self.multipliers[label] = label_multipliers[i]
```

```
def get_selected(self):
    """
        RETURNS:
            List of selected image identifiers
    """
    return list(self.count_selected_images +
                self.score_selected_images)[:]

def do_selection(self, image, image_name):
    """
        This method is used to do selection on images. The method
        saves selected image identifiers to class members. The
        selected images can be queried with get_selected() method.
        The reset() method canbe used to restart the selection
        done by this method.

        PARAMS:
            image: Next image to process as numpy array in RBG format
            image_name: Image identifier as string
    """
    # Do inference with YOLOv5
    results = self.model([image], size=640)

    total_count = 0
    average_score = 0

    names = results.names
    # Loop through detections and calculate count and score
    for i in range(results.n):
        for j in range(len(results.xyxy[i])):

            label = names[int(results.xyxy[i][j][-1])]
            score = results.xyxy[i][j][-2].item()

            if label in self.selected_labels:
                total_count += 1 * self.multipliers[label]
                average_score += score

    # Calculate average score
    if total_count > 0:
        average_score /= total_count
    else:
        average_score = 1.0

    if total_count > self.lowest_count:
        # Copy the score and name of the now replaced image
        replaced_image_name =
                    self.count_selected_images[self.lowest_count_index]
        replaced_image_score =
                    self.count_image_scores[self.lowest_count_index]

        # Save the new images information
        self.selected_image_counts[self.lowest_count_index] = total_count
        self.count_selected_images[self.lowest_count_index] = image_name
        self.count_image_scores[self.lowest_count_index] = average_score

        # Get new lowest count and list index
        self.lowest_count_index =
                    self.count_selected.index(min(self.count_selected))
```

```
            self.lowest_count = self.count_selected[self.lowest_count_index]

            # Set the replaced image's score and name for score
            # selection check
            average_score = replaced_image_score
            image_name = replaced_image_name

        if average_score < self.highest_score:
            # Save the new images information
            self.selected_image_scores[self.highest_score_index] =
                                                            average_score
            self.score_selected_images[self.highest_score_index] = image_name

            # Get new highest score and list index
            self.highest_score_index =
            self.selected_image_scores.index(max(self.selected_image_scores))
            self.highest_score =
                        self.selected_image_scores[self.highest_score_index]
```

APPENDIX B: EXPERIMENTATION RESULTS

Preliminary Experiment

	Person AP	Bicycle AP	Car AP	Bus AP	Truck AP
Had fine-tuned	0.024	0.347	0.878	0.047	0.172
Had pretrained	0.018	0.096	0.782	0.756	0.273
Has fine-tuned	0.137	0.440	0.911	0.033	0.738
Has pretrained	0.054	0.091	0.816	0.262	0.691
Hjo fine-tuned	0.203	0.077	0.889	0.841	0.427
Hjo pretrained	0.314	0.000	0.492	0.127	0.081
Ost fine-tuned	0.399	0.543	0.814	0.391	0.062
Ost pretrained	0.001	0.160	0.510	0.570	0.015
Rin fine-tuned	0.102	0.002	0.534	0.132	
Rin pretrained	0.343	0.000	0.282	0.097	

Experiment 1 Second Run with Modified Data

	Car AP	Bus AP	Truck AP
Had fine-tuned	0.895	0.082	0.117
Had pretrained	0.798	0.757	0.279
Has fine-tuned	0.923	0.084	0.751
Has pretrained	0.827	0.262	0.691
Hjo fine-tuned	0.924	0.743	0.439
Hjo pretrained	0.549	0.127	0.100
Ost fine-tuned	0.902	0.470	0.081
Ost pretrained	0.703	0.570	0.015
Rin fine-tuned	0.662	0.102	
Rin pretrained	0.327	0.097	

Experiment 1 Rerun with Fewer Images

AAU RainSnow, Has camera average results from 5 runs.

Random	Car AP	Bus AP	Truck AP	Selected	Car AP	Bus AP	Truck AP
10	0.743	0.008	0.146		0.723	0.013	0.193
30	0.869	0.017	0.624		0.874	0.047	0.656
50	0.900	0.097	0.611		0.903	0.047	0.718
75	0.918	0.046	0.720		0.917	0.078	0.765
100	0.928	0.074	0.805		0.925	0.215	0.756

AAU RainSnow, Hjo camera average results from 5 runs.

Random	Car AP	Bus AP	Truck AP	Selected	Car AP	Bus AP	Truck AP
10	0.614	0.029	0.009		0.279	0.001	0.010
30	0.824	0.025	0.072		0.776	0.293	0.029
50	0.853	0.585	0.028		0.823	0.307	0.019
75	0.867	0.720	0.130		0.858	0.626	0.094
100	0.886	0.702	0.132		0.890	0.702	0.068

AAU RainSnow, Had camera results.

Random	Car AP	Bus AP	Truck AP	Selected	Car AP	Bus AP	Truck AP
10	0.629	0.039	0.019		0.706	0.038	0.029
30	0.791	0.038	0.056		0.849	0.187	0.109
50	0.866	0.277	0.198		0.872	0.185	0.073
75	0.870	0.198	0.135		0.903	0.064	0.125
100	0.895	0.056	0.064				

AAU RainSnow, Ost camera results.

Random	Car AP	Bus AP	Truck AP	Selected	Car AP	Bus AP	Truck AP
10	0.690	0.157	0.027		0.654	0.256	0.023
30	0.810	0.176	0.043		0.847	0.450	0.057
50	0.896	0.838	0.082		0.909	0.621	0.060
75	0.920	0.625	0.053		0.892	0.614	0.067
100	0.874	0.730	0.129				

AAU RainSnow, Rin camera results.

Random	Car AP	Bus AP	Selected	Car AP	Bus AP
10	0.041	0.000		0.170	0
30	0.390	0.095		0.501	0.025
50	0.473	0.096		0.503	0.070
75	0.543	0.013		0.479	0.096
100	0.551	0.054			

Experiment 2

AAU RainSnow, car AP results with 1 to 4 camera training set.

	Had	Has	Hjo	Ost	Rin
1 camera	0.875	0.864	0.695	0.495	0.600
2 cameras	0.875	0.872	0.780	0.783	0.637
3 cameras	0.907	0.881	0.807	0.795	0.737
4 cameras	0.904	0.912	0.780	0.836	0.748

AAU RainSnow, bus AP results with 1 to 4 camera training set.

	Had	Has	Hjo	Ost	Rin
1 camera	0.525	0.301	0.222	0.054	0.005
2 cameras	0.680	0.209	0.288	0.162	0.001
3 cameras	0.743	0.188	0.340	0.254	0.033
4 cameras	0.748	0.227	0.454	0.439	0.081

AAU RainSnow, truck AP results with 1 to 4 camera training set.

	Had	Has	Hjo	Ost	Rin
1 camera	0.169	0.496	0.199	0.051	0.036
2 cameras	0.257	0.511	0.248	0.242	0.116
3 cameras	0.543	0.517	0.269	0.298	0.141
4 cameras	0.502	0.687	0.397	0.374	0.186

Experiment 3

AAU RainSnow, average results from all 5 runs with all cameras.

Random	Car AP	Bus AP	Truck AP	Selected	Car AP	Bus AP	Truck AP
25	0.724	0.069	0.142		0.737	0.052	0.093
50	0.833	0.326	0.355		0.833	0.435	0.221
75	0.856	0.342	0.295		0.861	0.524	0.342
100	0.813	0.538	0.383		0.870	0.565	0.421
125	0.867	0.516	0.408		0.879	0.541	0.541
Pretrained	0.705	0.528	0.464				

Experiment 4

DAWN dataset, rain category average results from all 5 runs.

	Person AP	Car AP	Bus AP	Truck AP
Random 10	0.004	0.659	0.034	0.107
Random 30	0.113	0.838	0.219	0.514
Random 50	0.403	0.875	0.093	0.686
Random 75	0.449	0.857	0.312	0.616
Random 100	0.602	0.899	0.425	0.647
Pretrained	0.735	0.735	0.221	0.457
Selected 10	0.001	0.696	0.113	0.261
Selected 30	0.005	0.826	0.136	0.514
Selected 50	0.205	0.872	0.177	0.674
Selected 75	0.549	0.895	0.231	0.688
Selected 100	0.616	0.895	0.318	0.671

DAWN dataset, fog category results.

	Person AP	Car AP	Bus AP	Truck AP
Random 10	0.010	0.628	0.254	0.018
Random 30	0.495	0.796	0.382	0.196
Random 50	0.609	0.882	0.527	0.381
Random 75	0.582	0.888	0.583	0.375
Random 100	0.761	0.902	0.662	0.430
Pretrained	0.784	0.806	0.754	0.291
Selected 10	0.002	0.685	0.251	0.107
Selected 30	0.028	0.851	0.441	0.246
Selected 50	0.705	0.898	0.428	0.387
Selected 75	0.724	0.893	0.646	0.504
Selected 100	0.805	0.911	0.712	0.512

DAWN dataset, sand category results.

	Person AP	Car AP	Bus AP	Truck AP
Random 10	0.011	0.676	0.014	0.025
Random 30	0.489	0.809	0.134	0.250
Random 50	0.678	0.842	0.199	0.257
Random 75	0.710	0.864	0.313	0.364
Random 100	0.696	0.880	0.380	0.486
Pretrained	0.738	0.738	0.674	0.365
Selected 10	0.007	0.567	0.033	0.017
Selected 30	0.225	0.799	0.327	0.190
Selected 50	0.553	0.848	0.366	0.301
Selected 75	0.773	0.867	0.349	0.395
Selected 100	0.740	0.888	0.371	0.437

DAWN dataset, snow category results.

	Person AP	Car AP	Bus AP	Truck AP
Random 10	0.203	0.497	0.047	0.036
Random 30	0.416	0.757	0.042	0.057
Random 50	0.590	0.814	0.513	0.239
Random 75	0.737	0.811	0.277	0.447
Random 100	0.711	0.827	0.712	0.546
Pretrained	0.760	0.767	0.572	0.446
Selected 10	0.111	0.603	0.023	0.062
Selected 30	0.523	0.778	0.086	0.310
Selected 50	0.641	0.828	0.268	0.384
Selected 75	0.644	0.820	0.223	0.465
Selected 100	0.692	0.807	0.446	0.495

Experiment 5

DAWN dataset, average results from all 5 runs with all categories.

	Person AP	Car AP	Bus AP	Truck AP
Random 20	0.088	0.695	0.072	0.128
Random 40	0.541	0.808	0.222	0.258
Random 60	0.597	0.830	0.245	0.391
Random 80	0.646	0.851	0.416	0.407
Random 100	0.678	0.852	0.442	0.468
Pretrained	0.740	0.796	0.631	0.382
Selected 20	0.048	0.693	0.097	0.199
Selected 40	0.438	0.822	0.320	0.339
Selected 60	0.631	0.844	0.257	0.422
Selected 80	0.642	0.848	0.495	0.473
Selected 100	0.666	0.860	0.532	0.461