CHALMERS
UNIVERSITY OF TECHNOLOGY

Detection of Dysplasia in Chronic Colitis using Neural Networks

Master's thesis in Biomedical Engineering

Tobias Alhede
Frida Nordlund

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021

Detection of Dysplasia in Chronic Colitis using Neural Networks

TOBIAS ALHEDE
FRIDA NORDLUND

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Signal processing and Biomedical engineering
Computer Vision and Medical Image Analysis
Chalmers University of Technology
Gothenburg, Sweden 2021

Detection of Dysplasia in Chronic Colitis using Neural Networks
TOBIAS ALHEDE
FRIDA NORDLUND

Supervisors:
Roman Naeem, Department of Electrical Engineering
Thomas De Lange, Sahlgrenska University Hospital

Examiner:
Fredrik Kahl, Department of Electrical Engineering

Master's Thesis 2021
Department of Electrical Engineering
Division of Signal processing and Biomedical engineering
Computer Vision and Medical Image Analysis
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Result from the created dataset where RetinaNET detected and predicted a dysplasia region with a probability score equal to 69%.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Detection of Dysplasia in Chronic Colitis using Neural Networks
TOBIAS ALHEDE
FRIDA NORDLUND
Department of Electrical Engineering
Chalmers University of Technology

Abstract

In recent years, Computer Aided Detection (CADe) has been widely used in medical images in order to increase the quality of diagnoses. This study focuses on using CADe for detecting cell changes in the form of dysplasia and hyperplasia in patients diagnosed with ulcerative colitis. A new dataset with manual annotation for these cell changes has been created, which includes images from ten colonoscopy examinations. We test the object detection models in combination with different augmentation techniques on this new dataset as well as on the existing KvasirSEG dataset [1]. KvasirSEG is an existing dataset containing 1000 gastrointestinal polyp images, annotated with both semantic segmentation and bounding boxes. KvasirSEG, as well as the new dataset, are retrieved during colonoscopy surveillance, which makes the KvasirSEG suitable for a pre-study. The best performing model with augmentation on the KvasirSEG dataset is VarifocalNet (VFNet) with a mAP score of 80.6%. For the new dataset, the best performing models are RetinaNET and VFNet in combination with different augmentation techniques with mAP scores of 9.3% and 8.7% respectively. The new dataset is lacking in quality since it contains several images from the same lesions, in combination with only subtle changes in the labelled region compared with the surroundings. In the present situation, we cannot determine the potential of this material, further work is needed before the dataset can be evaluated properly for object detection purposes.

Keywords: Dysplasia, Faster R-CNN, Hyperplasia, Inflammation, mAP, Object Detection, Polyp, RetinaNet, Ulcerative Colitis, VarifocalNet

Acknowledgements

A special thank you to our supervisor Roman Naeem, the Electrical Engineering department at Chalmers University of Technology. We would like to give about greatest gratitude for your supervision and help during our thesis, with everything from technical support to feedback on the report. Also, we wish to express our appreciation to our examiner Fredrik Kahl, for accepting the role and for helping us along the way.

The collaboration and great knowledge exchange with Olle Mannheimer has been greatly appreciated. Further, we are also grateful to Thomas De Lange for our medical supervision and to Tor-Jan Derek Berstad for all the technical help surrounding the dataset.

Lastly, a big thank you to everyone involved in the project from Chalmers and the other medical experts, who have helped and contributed to making this master thesis possible.

Thank you!

Tobias Alhede & Frida Nordlund, Gothenburg, June 2021

Contents

List of Figures

List of Tables

1
Introduction

In recent years, computer vision and more specifically image processing and pattern recognition has facilitated the medical field considerably [2]. It can be used to analyze medical images which can make it easier for the medical staff to make decisions in critical situations [2]. Since the computer vision systems are becoming fast and accurate, the diagnosis can also be determined faster and more accurately compared to previous solutions [2]. In this section, we are presenting the medical and technical background as well as our aim, limitations and specification of issue. Finally, we are also discussing the ethical considerations.

1.1 Medical Background

Chronic colitis and inflammatory bowel disease (IBD) are wide terms that describe different disorders in the gastrointestinal tract (GI-tract) characterized by chronic inflammation in the mucosa [3]. IBD includes two major disorders namely, Crohn's disease (CD) and ulcerative colitis (UC) [3]. CD is an inflammation in the patient's GI-tract that can be located somewhere from the oral cavity to the rectum [4], but often localized in the small or large intestine [5]. Since CD can be located in a wide range of locations, the symptoms can be quite various [4]. However, common symptoms of CD are abdominal pain, vomitting, diarrhea, constipation and, weight loss or weight gain [4]. UC is a disorder causing various degree of inflammation in the patient's large intestine or rectum characterized by relapses and remission [6], [7]. These two disorders have in common, an increased likelihood of the patient getting colorectal cancer in the future [8], [9]. In 2018, colorectal cancer was the third most common cancer type, more specifically around 10.2% of the total cases [10]. It was also the second most deadly cancer type accounting for 9.2% of the total cancer deaths registered [10].

Today, colonoscopy surveillance is recommended in order to detect changes in the colon that can result in a dangerous outcome [11]. An example of abnormal growth of cells is dysplasia, a precursor of cancer [12]. If these changes are detected early by colonoscopy surveillance, the risks of getting serious consequences will decrease [12]. However, colonoscopy disease detection miss-rates are too high, and articles mention that the miss-rate is somewhere between 17% and 28% in order to detect colorectal polyps [13]. As described earlier, this can result in development of colorectal cancer if the precursors are not found [13]. Therefore, it is of great importance to find different methods that can be used as a complement to the physicians in the area,

e.g. endoscopists, in order to detect the abnormalities of interest.

1.1.1 Technical Background

In order to analyse complex medical images recorded during colonoscopy, in an effective way, and to be able to find dysplasia that even trained endoscopists miss, Artificial Intelligence (AI) solutions are needed. Various studies have been done in different medical areas, where it has been seen that computer aided detection (CADe) using AI and deep learning can improve detection in medical images [14], [15], [16].

Deep learning uses deep neural networks, which are built up by several layers, each containing several neurons connected to the following layers in different ways, depending on the choice of layers. The structure of the networks can be divided into visible and hidden layers. The visible layers are the input and output which in this case are pixels and numerical output values respectively. From the visible input layer the next hidden layers can detect abstract features that after several layers can result in a prediction of what is seen in the image [17].

There are different AI solutions and techniques available today that all focuses on different aspects. When working with images and computer vision, techniques like object detection and segmentation are commonly used. Object detection labels and localizes the object with a square, called bounding box [18], and lastly segmentation labels every pixel in the image [19].

Every layer with neurons have corresponding weights that influences the importance of their information when going through the network [17]. This means that for an entire network, a large number of weights affect the outcome. These weights are not set by hand but from training the network. The network is trained by giving it images to train on, where the coordinates of the object of interest are existing. The images are processed in the network and its output is compared with the ground truth coordinates. When the output and the ground truth differ, the different weights are updated in order to make a more accurate and better network [17].

A well performing network is trained on a large dataset in order to recognize many different aspects in the images. Today, there are no existing labeled dataset of endoscopic images or videos from patients with dysplasia in chronic colitis. This means that it is of great priority to finalize a dataset in order to develop an algorithm for detecting the different classes in the data.

1.1.2 Aim

The aim of this master thesis was to test and evaluate if deep neural networks could be used to detect dysplastic lesions in the colon in patients with UC. A dataset of images from endoscopic investigations on patients diagnosed with UC have been created and labeled in collaboration with a medical student in order to get data for the deep learning algorithm.

1.1.3 Limitations

In order to narrow down the scope, we have limited our technical methods to object detection algorithms. We have not considered other possible methods, e.g. semantic segmentation and image classification. The dataset have only been annotated for object detection with labels and bounding boxes. We have only used labels of the following different classes, and examples of those can be found in Appendix A.1:

- Hyperplasia (Polyp and Biopsy)
- Dysplasia (Polyp and Biopsy)
- Inflammation (Polyp and Biopsy)

We have chosen to use KvasirSEG (referred to as Kvasir) for a pre-study in parallel with the creation of the new dataset since it is also made of gastrointestinal polyp images [1].

1.1.4 Specification of Issue Under Investigation

Our aim is complemented with the following questions that we have focused on answering during our project:

- How well does already existing systems perform on the new dataset compared to existing datasets containing frames within the colon?
- How accurately will the existing systems perform on the augmented datasets, measured in Mean Average Precision?
- What will be the target precision for the algorithm to be used in practical applications as a decision aid?

1.1.5 Ethical Considerations

When collecting and using patient data it is very important that the data handling is done correctly to respect the privacy of the patients and follows The General Data Protection Regulation. In this case the video material has been collected at examinations made at Akershus University Hospital in Norway, and it has been anonymised when saved and added to the collection of videos. Our work did not interfere with the patient treatments since the data is anonymous and was processed retrospectively. Therefore, this project is considered ethical. If the project could contribute to implement this type of technology in the healthcare, the diagnostic methods for these diseases will be improved and the number of patients that get the right diagnosis in time could increase.

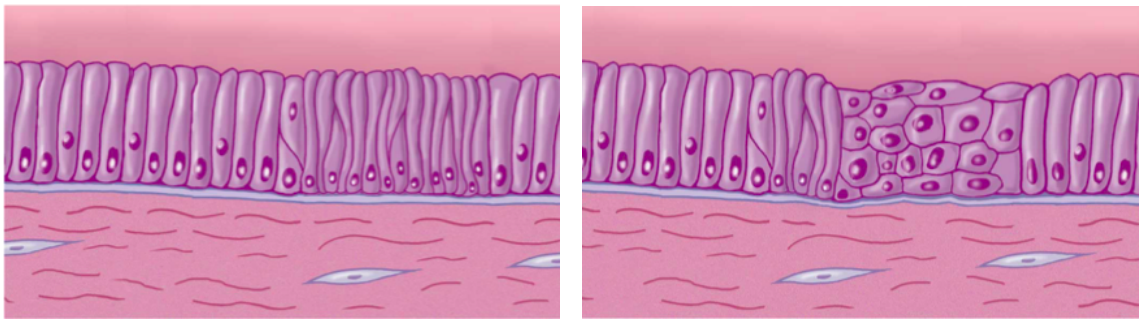
It is also important to mention the ethical discussion surrounding the use of AI today. There is always a risk that powerful technology meant for good purposes can be used with inappropriate intentions. With this in mind it is important that everyone reflects on the impact their solutions can have if used with wrong intentions. In our case we believe that the positive outcome of this thesis is larger than the risks.

2
Theory

In order to get a deeper knowledge in the area from both a medical and a technical perspective, the theory of the different parts are presented. To start with, the medical aspects are described, followed by the technical aspects in terms of object detection models. At last, the data augmentations techniques and the evaluation metric used in this thesis are explained.

2.1 Ulcerative Colitis and Cancer

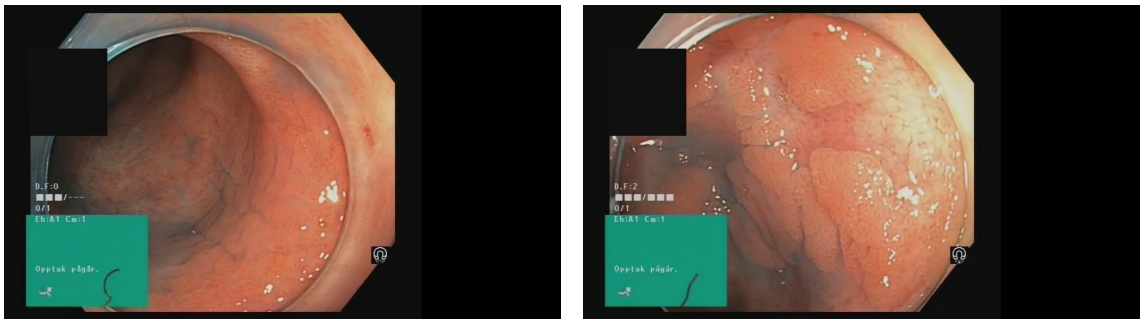
As earlier described, patients with UC are at increased risks for developing colorectal cancer in the future [20]. Surveillance of patients with UC are scheduled colonoscopy examinations in order to detect precursors of cancer as early as possible [20]. Lesions, an unhealthy mutation or change, can develop further into different phases of cancer [21]. Either, the lesions can stop developing at an unharmful stage and remain harmless in the future, or continue to develop into cancer [21]. The lesions can develop into non cancerous hyperplasia or the premalignant form, dysplasia [21]. In this project, the focus has been to detect dysplasia and differentiate it from hyperplasia and inflammation. In hyperplasia, the cells are increasing in numbers but they are still functional [21]. The cells in dysplasia have changed their microscopic appearance and may also loose their original functions [21]. An example of hyperplasia and dysplasia can be seen in Figure 2.1.



(a) Hyperplasia **(b)** Dysplasia

Figure 2.1: Two examples of the changes in the cell-structure in hyperplasia (2.1a) and dysplasia (2.1b). From [22].

The changes described above and in Figure 2.1 can also be visible in the mucosa of the colon. The structure of those changes can look like the examples in Figure 2.2 (example of images from the created dataset).



(a) Hyperplasia **(b)** Dysplasia

Figure 2.2: Examples of the mucosa in the colon of a patient with hyperplasia (2.2a) and/or dysplasia (2.2b).

2.2 Object Detection

As mentioned in the introduction, object detection is a field within computer vision where systems are used to localize objects of interest in images using bounding boxes, and identify each box using a label [18]. In order to predict labels and corresponding bounding boxes, networks need to be trained. Neural networks are explained more thoroughly in the following section.

2.2.1 Neural Networks

Today, there are various types of neural networks available. Perhaps the most well-known type is an artificial neural network (ANN), commonly described as neural networks [23]. The system is quite inspired by the function of the human visual cortex and its corresponding neurons, and more specifically how neurons are interconnected with each other in order to create a complex system [23]. The human nervous system starts with a signal containing a directive from the visual cortex. This signal travels through interconnected nerves to the final destination where the signal is represented by a visual command, for instance a movement of an arm or a leg [24]. In a similar way, neural networks are made of interconnected neurons, often called nodes, with non-linear activation functions and bias terms [23]. As described in the introduction, the network can be divided in three different sections, more specifically an input image, hidden layers and finally an output layer [23]. The neurons in the hidden layer are interconnected with each other, as well as connected with the input layer and the output layer respectively. In Figure 2.3, the methodology of a neuron in the hidden layer is visualised.

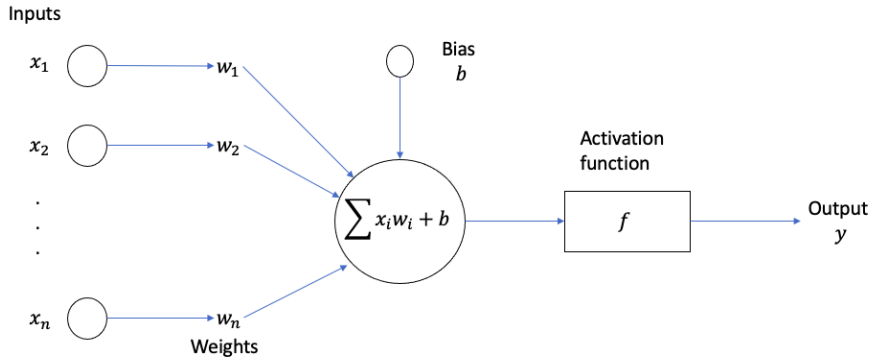


Figure 2.3: The methodology of the function of a neuron in a neural network, where x_i, $i = 1, ..., n$ are the input value from the previous layer of neurons, w_i, $i = 1, ..., n$ represent the weights, b is the bias term, f is the activation function and y is the output value to the next layer of neurons.

From Figure 2.3, it can be seen that the neuron in the current layer gets information from the neurons in the previous layer. The information from the previous layer is then multiplied by a specific weight for every neuron, which describes the importance of the information from the previous neuron [25]. When the network is trained, the weights are updated continuously according to their importance by forward- and backpropagation [25]. After the input values have been multiplied by their corresponding weight, a bias term may be added [23].

After this procedure, an activation function is used [17]. There are different types of activation functions used in the area, but one of the most frequently used is the rectified linear unit function, often shortened ReLU-function [17]. Its purpose is to turn on and off the neuron when the output reaches a specific threshold value [17]. The function is used in order to reassemble a biological neural network, where a specific threshold value equal to -55 mV needs to be reached to start an action potential in the biological nerve [24]. The ReLU-function is actually a non-linear function and its shape is represented in Figure 2.4.

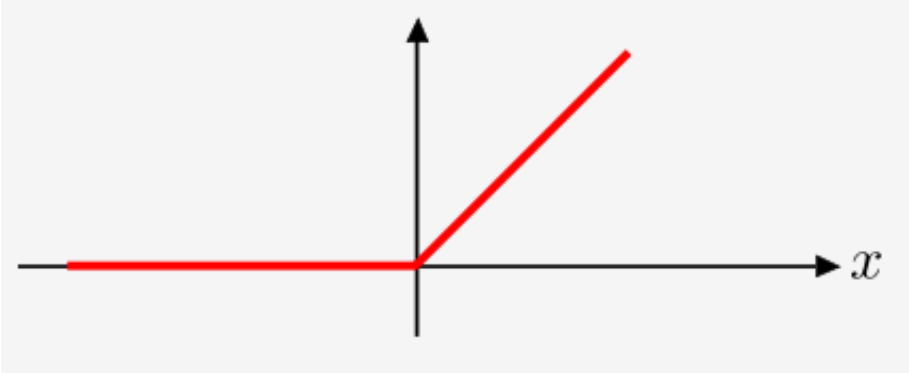


Figure 2.4: The shape of the ReLU-function. From [26].

If the input value is larger than a specific threshold value, the output value from the ReLU-function is expressed, with a non-zero value [17]. ReLU-functions are often

used either in the intermediate layers or at the end of the whole network used for the regression [23].

2.2.2 Convolutional Neural Networks

A class of deep neural networks often used in the image recognition area, and in particular object detection tasks, is the convolutional neural network (CNN). This class of networks reduces the number of neurons by mostly sharing weights between kernels, without reducing the deep knowledge of the whole system [27]. In turn, the training is computationally less expensive compared to other existing networks, *e.g.* fully connected neural networks [27]. The architecture of a typical CNN is visualised in Figure 2.5.

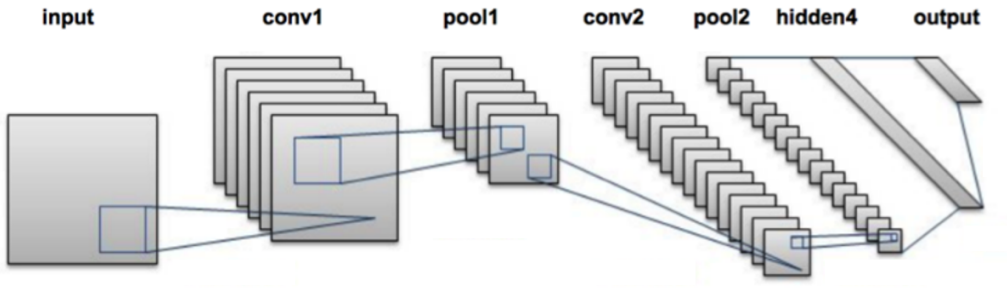


Figure 2.5: Architecture of a typical convolutional neural network, where the input image, the hidden layers and the output layer are represented. The hidden layer is represented by convolutional layers (conv), max pooling layers (pool) and the rest of the hidden layer (hidden) which often is made of more series of connected convolutional layers and max pooling layers. From [28].

As in Figure 2.5, the first layer in a CNN is the input layer. In image recognition, the input layer is often an image or a video frame with known information, *e.g.* certain width, height represented as pixel values and a specific number of channels. A gray-scale image has one channel, while a RGB-image is represented by three different channels that corresponds to the RGB color model. This means that the information of an image are pixel values represented as a 3D-array in memory [27].

2.2.2.1 Components in a Convolutional Neural Network

An input image is then connected to the first component in the hidden layer of a CNN. Often, the first layer is a convolutional layer which uses the mathematical operation named convolution on the input [27]. The convolutional layer is made of a filter, which in turn is represented by several kernels stacked on each other with specific sizes in form of a predefined width and height [27]. The filter is applied to a starting region (top left corner) on the input layer and then sliding above the entire image with a specific stride one step at a time [29]. Stride is the number of pixels the filter is moving at a time [29]. The size of the stride affects the dimension of the output layer. In Figure 2.6, an example of how a convolutional filter is applied to an input image is shown.

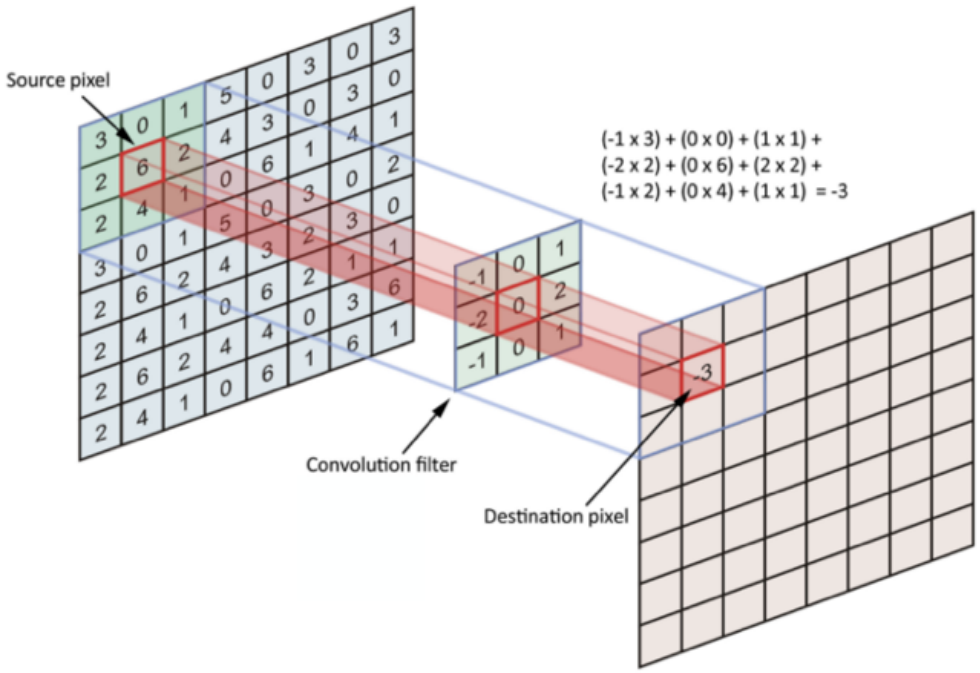


Figure 2.6: The operation and the result of a convolutional layer, which is made of a filter that uses element-wise multiplication. At last, these multiplications are summarised and present in the output layer. From [30].

In Figure 2.6, the input layer has the dimensions 8 × 8 pixels and the convolutional filter has the dimensions 3 × 3 pixels. At first, the filter multiplies each pixel value with its corresponding pixels in the input layer and then summarise the individual products. The size of the stride in combination with different padding techniques decide the output dimensions [27]. As a result of using CNNs, the number of neurons is fewer compared to systems where every neuron is connected to each neuron in the previous layer. However, the information of the corresponding area in a CNN is preserved since the abrupt changes in an image are remaining. To summarise, it can be seen that the behaviour of the convolutional filter in form of size, kernel values and stride have a great impact on the characteristics in the output layer.

Secondly, a very common layer in a CNN is the pooling layer which often follows a convolutional layer, or connected to a ReLU-layer in between the two layers [31]. Its function is to reduce the number of trainable parameters by downsampling the width and height of the previous layer which leads to a smaller size of the output layer [31]. Downsampling makes the resolution of the output layer lower but still containing large structural elements that often are of greater importance, while filtering smaller areas with finer details [31].

There are different types of pooling layers used in the area, but two of the most popular ones are the max-pooling layers and average pooling layers. Max-pooling layers calculate the maximum pixel value within each window of a feature map, which is the size of the kernel [31]. Average pooling layers operate in a similar way but instead calculate the average value within each feature map [31]. A max-pooling

layer with the kernel size equal to 2 × 2 pixels and with stride equal to 2 pixels is visualised in Figure 2.7.

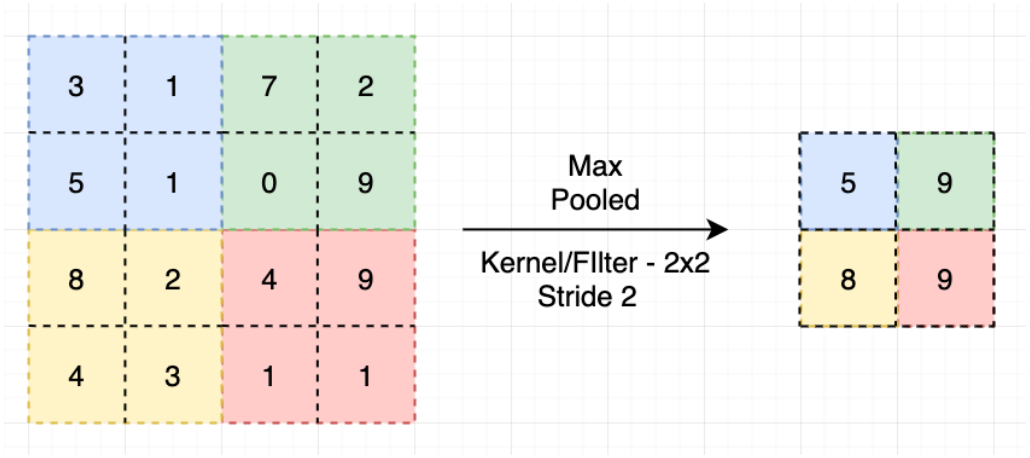


Figure 2.7: The methodology of how max-pooling layers operate. In this particular example, the kernel size is equal 2 × 2 pixels and stride equal to 2 pixels. The feature map windows where the max-pooling layer is operating are marked by various colors depending of the stride value. From [32].

In order to make appropriate settings on the network, it is important to understand how the size of the original input image is changing during the network path. The output dimensions after a specific layer can be calculated by using the following formula:

$$\frac{(V - R) + 2P}{S} + 1, \tag{2.1}$$

where V is the input size in one dimension, R is the kernel size in one dimension, P is the number of padding applied and S is the stride value [31].

The third layer, often located in front of the last convolutional layer is a fully-connected layer that can be used as a single element or in series with many fully-connected layers. This layer transforms the value of every neuron in the previous layer to generate a flattened array with output values [27]. Often, the output values are also normalised into a probability distribution using a softmax layer at the end of the network [27]. Depending of the purpose of the image recognition, the output array is designed differently. If the purpose is to give a class without a localization of the object, the size of the array is equal to the number of classes with element ranging from 0 to 1, where 1 corresponds to the highest probability for the respective class. For object detection, the output array also includes the coordinates of the bounding box in the final output array.

2.2.3 You Only Look Once (YOLO)

You Only Look Once (YOLO) is a family of object detection models that have shown good performance, while also focusing on improving inference speed. The basic idea behind YOLO is that it resizes the images, creates a grid (with size S $\times$ S) over the images and runs the detection on each of the grids at the same time. The image only needs to be processed once compared to other methods, using sliding windows or other more time consuming feature-extraction methods. We have chosen to focus on the third and the fourth version of YOLO.

YOLO's architecture is built up by a feature extractor and a detector [33]. The feature extractor used in YOLOv3 is Darknet-53 whose architecture can be seen in Figure 2.8, an improved extractor from the previous YOLO-models [34].

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 2.8: The structure of the Darknet-53 multi-scale feature extractor. From [35].

In Figure 2.8, it can be seen that Darknet is built of blocks with 2-stride convolutional layers in between that reduce the size of the data [34]. The blocks contain convolutional layers of 1 $\times$ 1 followed by 3 $\times$ 3 and a skip-connection (meaning that some layers are skipped in the backpropagation) [34]. The last three blocks, that down-scale using the 2-stride convolutional layers, create the grids on the different scales. These outputs are fed into the detector which looks at each grid-cell separately. Inside the grid-cells, pre-defined (using B boxes from K-mean clustering on the dataset) anchor boxes or bounding boxes are positioned on the centroid, regressed to correct size and finally classified by the detector [34].

YOLOv3 is a multi-scale detector, meaning that the detection on the images are done after scaling the images to different sizes [34]. Parts from the larger scales are also concatenated with the smaller scales, which means that large scale factors can influence the small scale results [34]. The detector produces a vector containing the bounding box´s relative position, the probability that the box contains an object which is called objectness score and also the probability for each of the classes to be found in the box [34]. This is calculated for every grid-cell, only keeping the best performing bounding boxes of each cell [34].

This network is trained using the following complex loss-function [34]:

$$\begin{aligned} L &= \lambda_{coord} \sum_{i=0}^{B} obj_i \times MSE((x_i, \hat{x}_i), (y_i, \hat{y}_i)) \\ &+ \lambda_{coord} \times \sum_{i=0}^{B} obj_i \times MSE((\sqrt{w_i}, \sqrt{\hat{w}_i}), (\sqrt{h_i}, \sqrt{\hat{h}_i})) \\ &+ \sum_{i=0}^{B} BCE(obj_i, \hat{obj}_i) \\ &+ \lambda_{no_{obj}} \sum_{i=0}^{B} BCE(objscr_i, \hat{objscr}_i) \times (1 - obj_i) \times ignore \\ &+ \sum_{i=0}^{B} BCE(C_i, \hat{C}_i) \times obj_i. \end{aligned} \tag{2.2}$$

In Equation (2.2), λ_{coord} and $\lambda_{no_{obj}}$ are hyperparameters used to balance the effect between bounding box coordinate precision and confidence predictions of boxes that does not contain any object. obj_i is the objectness score mentioned above, MSE stands for mean square error and BCE for binary cross entropy. x, $\hat{x}$, y and $\hat{y}$ are the coordinates of the centroid of the prediction ($\hat{x}$ and $\hat{y}$) and ground truth (x and y). w and h are the width and height of the bounding box. $ignore$ is a parameter used to minimize the importance of the boxes with a little difference in overlap.

It can be seen that the loss function in Equation (2.2) is calculated as the sum of several terms which in turn can be seen as separate losses. The first term represents the loss of the predicted centroid coordinates, the second describes the loss of shape of the predicted bounding box, the third and the fourth describes the objectness and non-objectness scores. Lastly, the classification score is added to create the entire loss-value.

The fourth version of YOLO is an improvement of the third version. The structure of YOLOv4 is built up of a backbone called CSPDarknet53, a neck of Spatial Pyramid Pooling (SSP) and Path Aggregation Network (PAN), and also the YOLOv3 head [36]. CSPDarknet53 is a Darknet53 backbone that applies the Cross Stage Partial Network (SCPNet) strategy [36]. SCPNet is a method to directly optimize the mapping function of the network, making it more efficient in handling the many concatenations used in a complex network [37]. Layers between the backbone and the head are called the neck of the model. In YOLOv4, the neck consists of SSP and PAN [36]. SSP is a method created to solve the problem of static input sizes using i.e fully connected layers and varying image sizes [38]. Fixed length representations are made by generating the feature map from the entire image once and then doing pooling (i.e max-pooling) on sub-images. Then, the output is a vector containing the feature map and the output of the pooling [38]. This is illustrated in Figure 2.9.

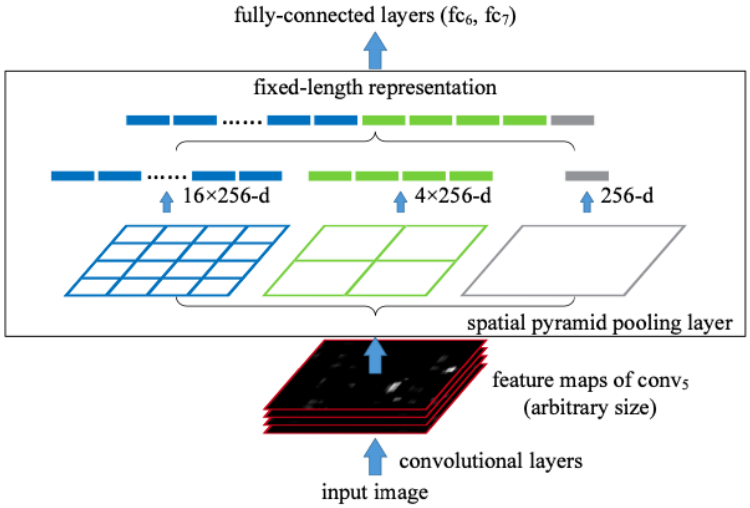


Figure 2.9: The structure of the spatial pyramid pooling layer. From [39].

As can be seen in Figure 2.9, the input size to the SSP-layer does not affect the output size which solves the problem of static input sizes. The next part of the neck is the PAN, a method for preserving the location information when the complexity of the convolutional features gets larger [40]. This is done by creating shortcuts between lower layers and the top layers, using features from all layers and making the network decide which are the best to use and lastly by using both fully connected and convolutional layers [40]. PAN is a further development of the Feature pyramid network, FPN. The structure of both FPN and PAN can be seen in Figure 2.10.

(a) FPN **(b)** PAN

Figure 2.10: The structure of FPN and PAN, where the P_i symbolizes the output from convolutional layers with the size of $\frac{1}{2^i}$ times the input image size. From [41]

FPN is a simple way to preserve and add features of different scales together. It is done by taking outputs from smaller (in scale) outputs and adding them to the larger scale outputs [42]. This is achieved by using a pyramid structure of convolutional layers and the natural resizing of the image that occurs in convolution, which is illustrated in Figure 2.11.

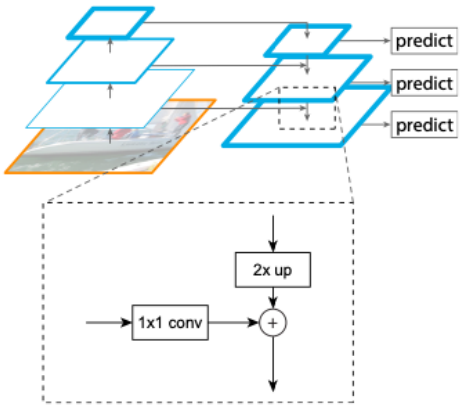


Figure 2.11: The pyramid structure of convolutional layers used in FPN, with an example of how the reshaping of the connections is done. From [43].

In Figure 2.11, the structure used is described with an example of three different scales. The input-image is scaled down by using convolutional networks in three layers which results in outputs from each of the three scales. These outputs are used to create one prediction, per layer [42]. Uniquely for FPN is that it is composed of both a top-down- and a bottom-up pathway. The bottom-up pathway is used for feature extraction which results in high resolution but semantically weakened feature maps, while top-down pathway is used to upsample higher resolution features to a coarser size containing semantically stronger feature maps [42]. The outputs are merged by lateral connections (which are quite similar to skip connections) to help the detector predict the region of interests better [42].

The further development of FPN to create PAN is done by also reversing this addition by adding a second bottom-up pathway, which is illustrated in Figure 2.10b. The outputs from the FPN-part are added to the output of the nodes from only the smaller scales, preserving the larger features in the smaller scales as well [44].

2.2.4 Regions with CNN features (R-CNN)

This is a algorithm named Regions with CNN features (R-CNN), which are a class of object detection techniques that uses two-stage detectors [45]. The first stage generates region proposals, which are regions in the image with high probability of containing valuable information, while the second stage detector uses the region proposals to classify the object within the region and also estimate coordinates for the bounding boxes [45]. In previous years, other detectors based on methods such as histogram of oriented gradients (HOG) and scale-invariant feature transform (SIFT) were used. These were the golden standard within the object detection area. However, R-CNN showed promising results with higher object detection performance compared to earlier systems [45].

The first version of the family of methods, named R-CNN, builds mainly on three different modules combined, namely the region proposal algorithm which proposes region of interests (RoI's), the feature extractor (backbone) which extracts features from the RoI's, and the classifier based on class specific linear support-vector machines (SVM's) which classifies each region [45]. The region proposals algorithm uses selective search in order to identify and extract around 2000 RoI's which will serve as a base for further investigations [45]. Selective search is a method used to localize the objects in an image and at the same time to save computational power compared to earlier methods that are quite time-consuming (*e.g.* sliding window of different sizes). In Figure 2.12, the different modules are visualised.

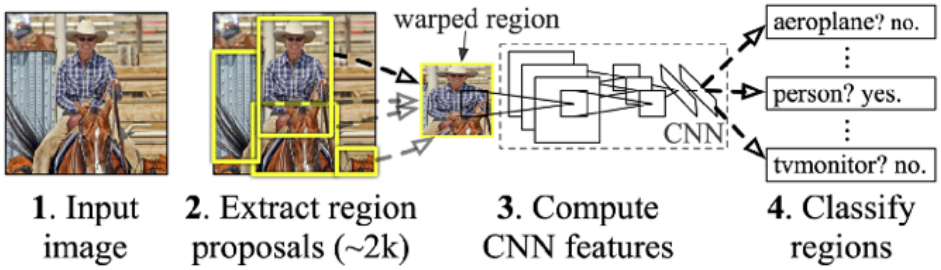


Figure 2.12: An overview of the system R-CNN. An input image is used to extract around 2000 region proposals. These region proposals are cropped, normalized and fed to a CNN where features are calculated. Finally, the objects in the regions are classified by using a specific linear SVM specific for each class. From [46].

In R-CNN, the selective search works in the following way. Firstly, around 2000 initial regions are created [47]. The next step is to use an algorithm in order to group neighboring regions together through an iterative process [48]. There are various methods used in order to group the regions, but some of the most common methods for this purpose are by checking colors, textures and surrounding pixel areas [48]. It is important to use different methods in this stage since neighboring pixels can be connected to each other in many different ways. The two most similar neighboring regions are grouped together, whereas new similarities are calculated between this groups and other regions until the original image is connected [48].

Next step in the procedure is the feature extraction by using CNN. The 2000 candidate region proposals are cropped to a specific size to work with the CNN, 227×227 pixels where all values are normalized as well [45]. These features are used as input values to a CNN made of five different convolutional layers followed by two fully connected layers, resulting in an output vector with size equal to 4096 elements [45].

Lastly, the output array is fed into a linear classifier in order to classify the objects in the regions [45]. The technique was changed in R-CNN from using softmax layers, which as decribed earlier is a function that gives the object a probability value from 0 to 1 describing the actual class, to a linear SVM layer specific for each class. The SVM classifer has scores for every class evaluated from test sets where non-maximum suppression is used in order to reject a specific region if it gets a

intersection-over-union (IoU) overlap with a neighboring region with a higher class-score [45]. Another aspect with the SVM is that R-CNN uses a specific threshold value equal to 0.3 to distinguish between positive examples and negative examples in the learning process [45]. Positive examples are images containing objects with a specific class, while negative examples are images containing other classes.

In order to make the method more precise in its localization of proposed bounding boxes, an optional bounding box regressor can be used. The regression model uses the predicted bounding box coordinates $\mathbf{p} = (p_x, p_y, p_w, p_h)$ in combination with the ground truth targets (t_x, t_y, t_w and t_h), where the indices x and y are the center coordinates of the bounding box, w is the width and h is the height to minimize a standardized least squares loss function [45]. The loss function is expressed as follows:

$$\ell_{reg} = \sum_{i \in (x,y,w,h)} (t_i - d_i(\mathbf{p}))^2 + \lambda ||\mathbf{w}||^2, \tag{2.3}$$

where λ is the regularization term equal to 1000 and $\mathbf{w}$ is a vector of learnable model parameters [45]. In order to minimise the loss function, the R-CNN regressor learns the scale-invariant transformation (defined as $d_i(\mathbf{p})$, where $i = x, y, w, h$) between the two center coordinates of the predicted box and the ground truth box, with the log-scale transformations between the different widths and heights of the bounding boxes [45].

2.2.5 Fast R-CNN

R-CNN is a technically good model for object detection in terms of accuracy, but the implementation also has some major drawbacks. Examples of drawbacks in the method are that it is very time- and memory-consuming since the method needs to classify around 2000 region for every single image, but also that real-time implementation is impossible since the evaluation procedures for one image takes around 47 seconds [49]. Therefore, the authors of R-CNN developed a faster algorithm named Fast R-CNN to eliminate some of the drawbacks with the R-CNN method. However, the two methods are quite similar. In Figure 2.13, a conceptual design of a Fast R-CNN technique is visualised.

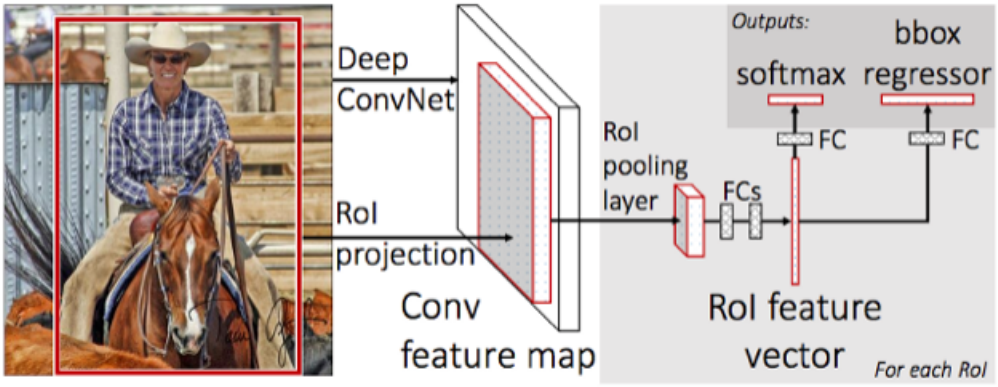


Figure 2.13: The architecture of the Fast R-CNN model, where an image is used as an input in combination with several regions of interest. The inputs are passed through a CNN, and the regions of interest are pooled to a feature map with specific size. After the fully-connected layers, the feature vector is fed to either a softmax layer that calculates the softmax probabilities values or to a bounding box regressor that calculates the offset values compared to the ground truth values using a specific loss function. From [50].

First of all, the Fast R-CNN uses the entire image as an input to a CNN in order to generate a convolutional feature map [49]. The CNN is made of several convolutional layers and max pooling layers, while one of the last element is a RoI pooling layer [49]. Each region proposal is pooled by the RoI-layer into a fixed-size feature vector from the feature map [49]. Then, the feature vectors are passed through some fully-connected layers which result in two different output vectors per RoI. One vector represents the softmax probability estimates per class, while the other vector represents the offset values of the bounding box coordinates for each class [49].

Fast R-CNN does not need a specific input size of the image, which was a criterion in the R-CNN method (more specifically 227×227 pixels). The Fast R-CNN solves this problem by using a RoI pooling layer, which divides the input region proposals into a grid with size equal to 7×7 [49]. This results in an output with a constant size since the RoI pooling layer uses a max-pooling layer in each grid of the input map [49].

Another improvement with the Fast-RCNN method is that it uses a softmax layer instead of the SVM layer in R-CNN, which leads to a faster process, thus even more accurate compared to R-CNN [49]. Since the Fast R-CNN model estimates the classes of the objects and corresponding bounding box coordinates, it also uses a different loss function that takes these parameters into account. The name of this function is multi-task loss on each labeled RoI [49], and is expressed as follows:

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v), \tag{2.4}$$

where p is a discrete probability distribution for each RoI, u is the ground truth class, t^u is the bounding box regression offset vector, v is the ground truth bounding box regression target vector and λ is a controlling parameter often set to 1 [49]. The

parameter $[u \geq 1]$ is defined the Iverson bracket indicator function:

$$[u \geq 1] = \begin{cases} 1, & \text{if } u \geq 1 \\ 0, & \text{otherwise.} \end{cases} \tag{2.5}$$

The first term in Equation (2.4) is defined as follows:

$$L_{cls}(p) = -\log(p_u), \tag{2.6}$$

where it is described as the log-loss for the ground truth class [49]. The second term in the loss function is described as:

$$\begin{gathered} L_{loc}(t^u, v) = \textstyle\sum_{i \in (x,y,w,h)} \text{smooth}_{L_1}(t_i^u - v_i) \\ \text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise} \end{cases} \end{gathered} \tag{2.7}$$

This term leads to that the loss function described in Equation (2.4) is not as sensitive to outliers compared to R-CNN, since this loss function has more terms that compensates for those deviations [49].

When those changes were implemented in the Fast R-CNN model, the accuracy of the method was slightly better compared with R-CNN, more specifically an increment with about 0.9% on a dataset named as PASCAL VOC 2007 test set (from R-CNN mAP 66.0% to Fast R-CNN mAP 66.9%) [49]. This is not a huge improvement of mAP, however the Fast R-CNN is much faster compared to its precursor. The estimated test rate for Fast R-CNN is about 0.32 seconds per image (the selective search algorithm which takes about 2 seconds is not included in this estimation). This can be compared with R-CNN which has the value 47 seconds per image, and the Fast-RCNN only needs to get the image as input once compared to the 2000 region proposals in R-CNN [49].

2.2.6 Faster R-CNN

Even if the Fast R-CNN is quite fast on test images and is near the potential to be used in real-time, it is still not possible with the actual technique. The reason behind this is that the selective search algorithm used in both R-CNN and Fast R-CNN is a quite time-consuming process [51]. Improvement of Fast R-CNN has been made that eliminates the selective search algorithm in order to make a more efficient model with respect to time and accuracy. This model is named as Faster R-CNN, and it uses a different technique in order to learn the different region proposals in an image [51]. The method that replaces the selective search algorithm is named Region Proposal Networks (RPN) [51]. This method is used to propose relevant regions in the image, while the second part of the model is the Fast R-CNN detector used to process the feature maps in combination with the region proposals [51]. The architecture of the model can be seen in Figure 2.14.

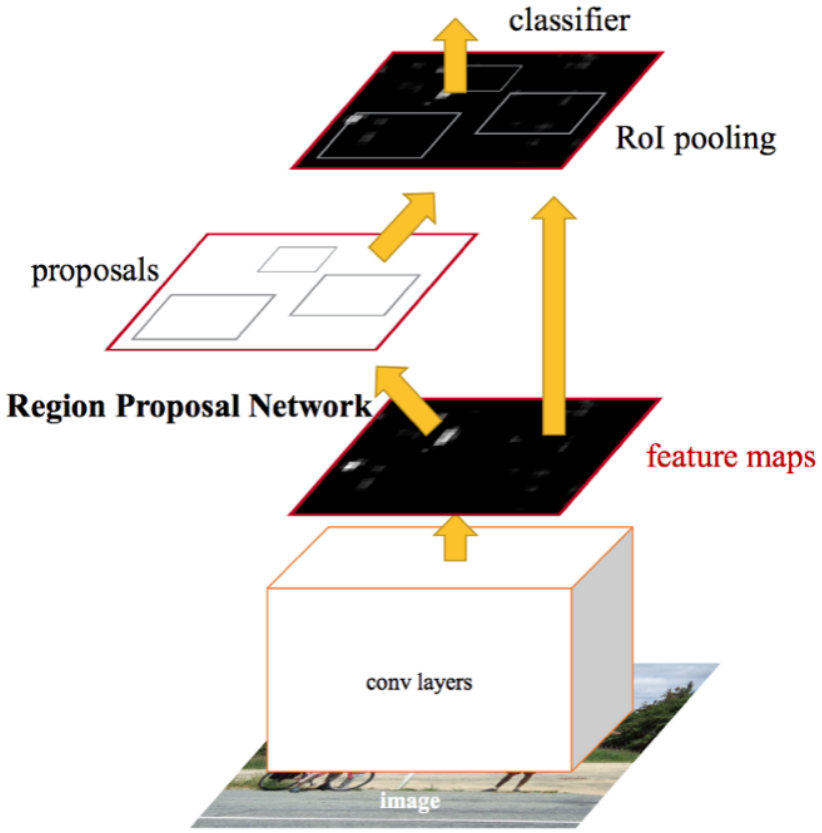


Figure 2.14: The architecture of the Faster R-CNN, where a RPN is used in order to detect relevant objects from the convolutional feature maps. From [52].

As can be seen in Figure 2.14, the system is a complex and connected network optimized for object detection. The model starts with a feature extractor where the image is the input in order to get the convolutional feature maps, which is the same methodology as in Fast R-CNN [51]. Then, the new method RPN is used. Its purpose is to tell the Fast R-CNN detector module where in the feature maps it should operate [51].

First of all, the RPN uses a smaller network over the input image which is expressed by its corresponding convolutional feature map [51]. The sliding window is in turn connected to an intermediate layer with a specific design and number of layers [51]. In Figure 2.15, a network with 256 layers is used. After the intermediate layer of the network, there are two similar fully-connected layers which are defined as the box-classification layer and a box-regression layer [51].

There is a specific methodology behind the spatial window technique in the Faster R-CNN model. At each location of the sliding window grid, the model predicts several region proposals at the same time as it operates [51]. In Figure 2.15, the maximum number of possible proposals for each sliding window grid is denoted by an integer k, which results in an output of the box-classification layer equal to $2 \times k$ scores that represents the probability values of an object or not for the proposals

[51]. The other fully-connected layer, the box-regression layer, has $4 \times k$ outputs values representing the coordinates of the proposals for each sliding window grid [51]. These proposals are called the anchor boxes, and is centered in each sliding window with a specific scale and aspect ratio [51], which can be seen in Figure 2.15.

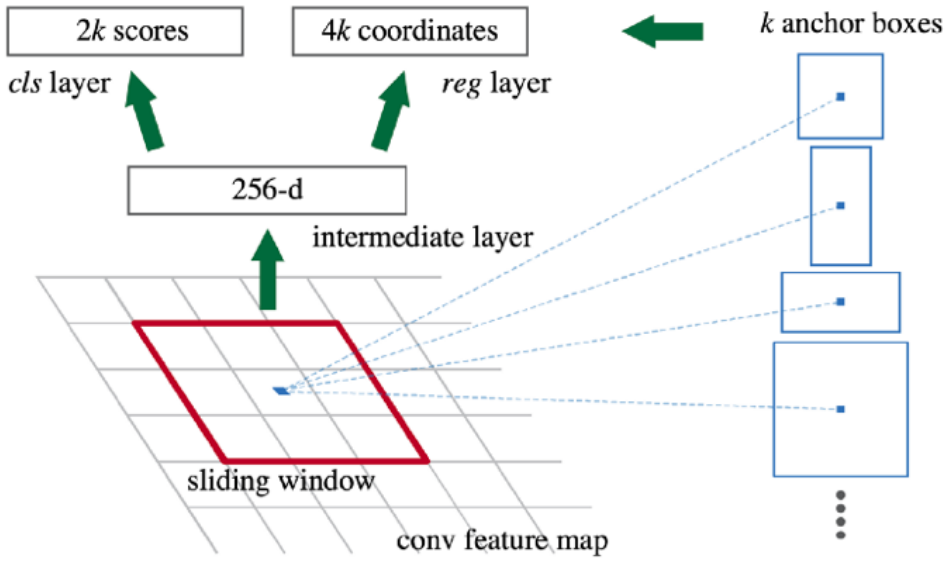


Figure 2.15: The methodology of how Region Proposal Networks (RPN) works, where the centered anchor boxes are visualised as well as a network designed with 256 layers. Finally, the two sibling layers at the end which classifies and locates the objects. From [41].

Another important aspect of this model is the translation-invariant definition, which guarantees that the model can predict the proposal in both the original image and in the anchors boxes independently of the location [51]. This in turn reduces the size of the RPN compared to earlier models, which results in much fewer parameters to learn and therefore less time-consuming [51].

In order to train the RPN, at least one of the two different criteria needs to be fulfilled to classify an anchor box with a positive label, which means an object of interest. The first criterion is that an anchor box has a positive label if its corresponding anchor box has the highest IoU value [51]. Secondly, the other criterion is defined as an anchor box with an IoU overlap higher than 0.7 with any other ground-truth box [51]. In the other case, the network also needs negative examples to train on. A non-positive anchor will get a negative label if its corresponding anchor box has an IoU value lower than 0.3 for all the ground-truth boxes [51]. In the third case, when the anchor boxes neither fulfil the positive label criteria nor the negative label criterion, the boxes will be discarded and not be used to train the RPN with [51].

A specific loss function is used in the Faster R-CNN model that minimizes during the training procedures. More specifically, the loss function is a multi-task loss similar to the function used in Fast R-CNN [51]. The loss function for Faster R-CNN

is presented as follows:

$$L(p_i, t_i, p_i^t, t_i^t) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^t) + \lambda \frac{1}{N_{reg}} \sum_i p_i^t L_{reg}(t_i, t_i^t), \tag{2.8}$$

where i is the index of the anchor, p is the predicted probability of an object, p^t is the ground truth (equal to 1 if the anchor is positive, 0 if the anchor is negative), t is the coordinates of the predicted bounding box and t^t is the ground-truth coordinates [51]. N_{cls} and N_{reg} are two normalizing factors, while λ is a balancing hyperparameter default to 10 by the creators [51].

The method can also be trained with images of various sizes and ratios since the last layer in the RPN network is made of a convolutional layer instead of a normal fully-connected layer [51]. As mentioned earlier, the Faster R-CNN uses the same technique as Fast R-CNN in the detector. The difference is that Faster R-CNN gives the detector locations where it should operate [51]. This leads to that the system can nearly be used for real-time purposes, with a frame rate equal to 5 frames per second, and also a higher accuracy compared to its precursors [51].

2.2.7 EfficentDet

EfficientDet is created as an efficient, accurate and scalable multi-scale detection architecture [44]. It consists of the backbone EfficientNet, repeated Weighted Bi-directional Feature Pyramid Network (BiFPN), class prediction network and at last box prediction network [44]. The structure of the networks can be seen in Figure 2.16.

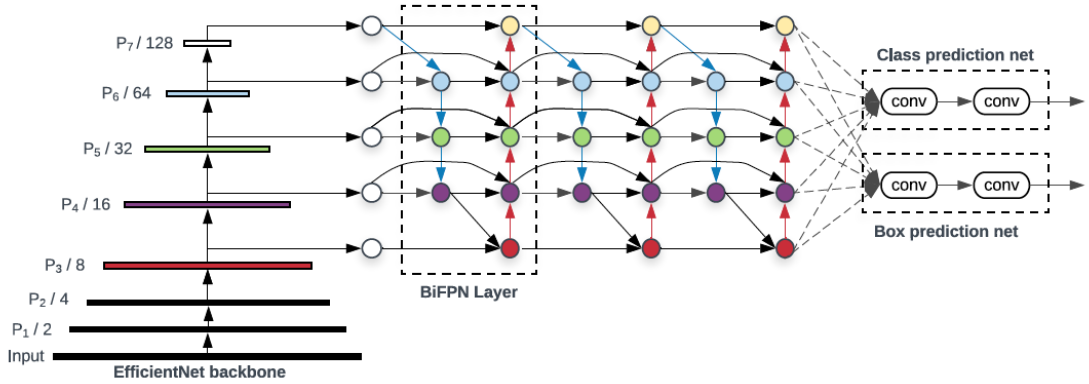


Figure 2.16: The structure of EfficientDet, including the backbone EfficientNet, repeated Weighted Bi-directional Feature Pyramid Network (BiFPN), a class prediction network and a box prediction network. From [53].

The architecture backbone, EfficientNet uses a baseline that can be seen in Figure 2.17 and is a network created to renew the process of scaling up CNNs [54].

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Figure 2.17: The structure of the backbone in EfficientNet. From [53].

A network can normally be scaled in three ways; the width (number of channels per layer), the depth (number of layers) or the resolution of the image [54]. In EfficientNet, compound scaling is presented that scales all these three parameters with fixed set of coefficients [44]. This is justified with the argument that if the image is bigger, more layers will be needed as well as more channels to see the smaller patterns [54]. The scaling model for the parameters depth (d), width (w) and resolution (r) is presented as:

$$\begin{aligned} d &= \alpha^\phi \\ w &= \beta^\phi \\ r &= \gamma^\phi \end{aligned} \tag{2.9}$$

Here, the coefficients are scaled uniformly using a compound coefficient ϕ [54]. The coefficients α, β and γ are positive integers larger than one and the networks' floating-point operations per second (FLOPS), defined as proportional to d, w^2, r^2 that should be approximately 2^ϕ. Therefore the two conditions:

$$\begin{aligned} &\alpha \times \beta^2 \times \gamma^2 \approx 2 \\ &\alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{2.10}$$

should be fulfilled [54]. EfficentDet introduced the BiFPN, which is a further development of FPN [44]. Earlier FPN-methods, like for example PAN mentioned earlier, have been developed to fuse input features of different kinds [40]. The problems of those earlier methods have mostly been that the features of different scales contribute to a different extent and only using summation would ignore that [44]. The creators of EfficientDet states that PAN performs better than the original FPN in terms of accuracy while being more costly in computational power [44]. A way to both introduce a weighted method to sum up the different scaled features and to minimize the computational power is therefore investigated [44]. The computational power and the number of parameters are minimised by removing nodes

(connections) with only one input and then an additional connection between the input and output of the same sizes are added for the improved precision [44]. The structure of BiFPN can be seen in Figure 2.18.

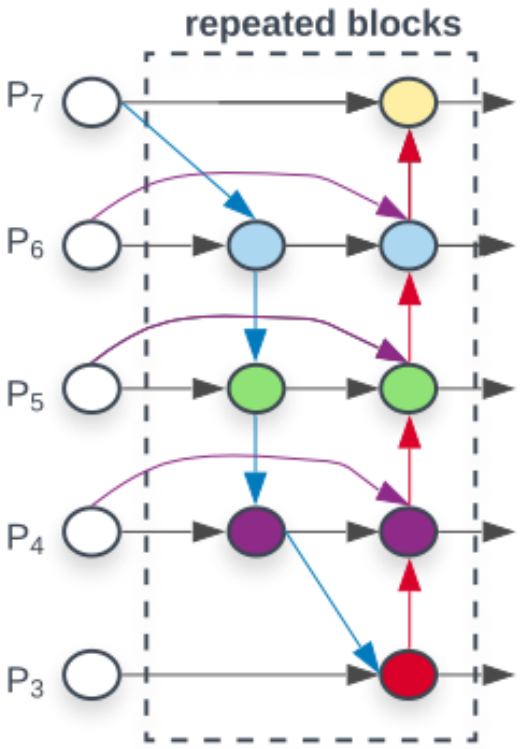


Figure 2.18: The structure of BiFPN, where the P_i symbolizes the output from a convolutional layer with a size of $\frac{1}{2^i}$ times the input image size. From [55].

Another update to this network is the weights used when fusing different features [44]. Different features are differently important and the network itself learns which weights that are more important than other [44]. EfficientDet scales just like EfficientNet, the entire model with a coefficient ϕ [44]. The scaling includes the backbone, BiFPN, class/box network and the image resolution [44]. The scaling of the backbone is done with the same method as in EfficientNet [44]. The depth of the BiFPN networks D_{BiFPN} is linearly increased and the width W_{BiFPN} is increased exponentially as is follows [44]:

$$\begin{aligned} D_{BiFPN} &= 3 + \phi \\ W_{BiFPN} &= 64 + 1,35^{\phi}. \end{aligned} \tag{2.11}$$

The width of the box and class predictor is set to equal the the width of the BiFPN and the depth of the predictors D_{Box} and D_{Class} are scaled according to [44]:

$$D_{Class} = D_{Box} = 3 + \frac{\phi}{3} \tag{2.12}$$

The resolution of the image is scaled linearly according to the following formula:

$$R_{input} = 512 + 128\phi. \tag{2.13}$$

2.2.8 RetinaNET

RetinaNET is an object detection model built on a single-stage detector [56]. It uses techniques similar to RPN in the Faster R-CNN in combination with features pyramids as in FPN (described in section 2.2.3) [56]. Further, this method uses a new designed loss function which has shown great potential with respect to accuracy [56]. The architecture of RetinaNET is visualised in Figure 2.19.

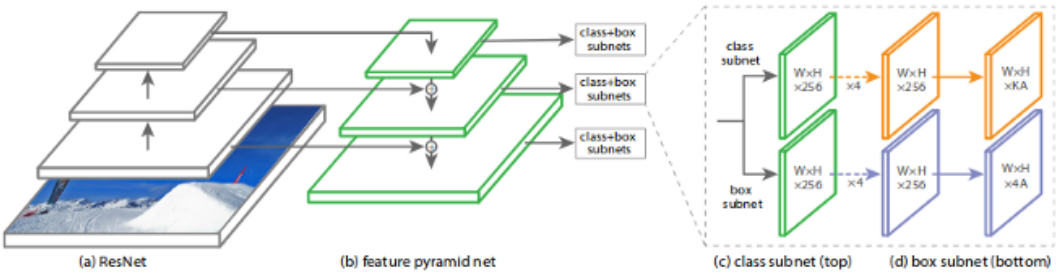


Figure 2.19: The architecture of the RetinaNET. A Feature Pyramid Network (FPN) backbone is used on top of a ResNet architecture which is connected to two different subnetworks that predict classes from the anchor boxes and estimate the bounding box coordinates using regression with the ground-truth boxes. From [57].

To start with, the creators have developed a new loss function named Focal Loss [56]. The Focal Loss is implemented in order to compensate if there is an imbalance between foreground classes and background classes during the training procedure [56]. The binary cross entropy (CE) loss for binary classification is presented below, as well as the Focal Loss:

$$CE(p, y) = CE(p_t) = -\log(p_t), \tag{2.14}$$

and

$$p_t = \begin{cases} p, & \text{if } y = 1 \\ 1 - p, & \text{otherwise,} \end{cases} \tag{2.15}$$

where $y \in (\pm 1)$ is the ground-truth class (equal to $+1$ for the ground truth class and otherwise -1), $p \in [0, 1]$ is the estimated probability for the correct class (which means $y = 1$) and p_t is used to simplify the overall derivation [56]. The creators are making this derivation for the binary classification, but they also claim that it can be quite easily extended to multi-classes [56]. In order to compensate for class imbalances, Equation (2.14) can be modified by adding either a weighting factor $\alpha \in [0, 1]$ if $y = 1$ or $1 - \alpha$ if $y = -1$ [56]. From this information, Equation (2.14) can be modified to the following expressions:

$$CE(p_t) = -\alpha_t \log(p_t) \tag{2.16}$$

and

$$\alpha_t = \begin{cases} \alpha, & \text{if } y = 1 \\ 1 - \alpha, & \text{otherwise.} \end{cases} \tag{2.17}$$

The creators who developed RetinaNET uses Equation (2.16) and (2.17) as a baseline to the new loss function. They showed in various experimental results that the existing loss functions described in Equation (2.16) can handle the importance of positive and negative examples by the weighting factor α, but lack the possibility to differentiate between easy and harder examples [56]. They have solved this by adding a parameter which they called a modulating factor including a focusing parameter (γ). The Focal Loss can be seen below:

$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t), \tag{2.18}$$

where γ is set to 2 and α is set to 0.25 for best results in form of minimizing the loss function according to results the creators presented in their publication [56].

As mentioned earlier, the RetinaNET detector is actually a single-stage method which is built on a backbone network and two different subnetworks specific for the classification and the bounding box regression [56]. Both subnetworks are applied on the output from the backbone network, which is a convolutional feature map [56]. The FPN described in Figure 2.10a is used in combination with the network ResNet to create the backbone network [42]. This is used to get various scales of the original input image which lead to that objects at different scales can be detected by the model [56]. Anchors are also used in this model similar to the ones in the RPN used in Faster R-CNN, with 5 different pyramid levels with increasing area from 32^2 to 512^2 pixels [56]. In RetinaNET, three different aspect ratios and anchor sizes are used, more specifically {1:2, 1:1, 2:1} and $\{1, 2^{1/3}, 2^{2/3}\}$ which results in 9 different anchors [56].

The classification subnetwork predicts an estimated probability of an object for each of the anchors and the object classes [56]. The architecture of this part is also a FPN but uses four following convolutional layers with size 3 $\times$ 3 pixels each designed with 256 filters, and followed by ReLU-functions. This in turn is followed by an additional convolutional layer with the same size but with the number of filters dependent of the product between the number of classes and number of anchors [56].

The last component in the design is the bounding box regression subnetwork, which has similar design as the classification subnetwork but instead predicts the bounding box coordinates and regresses the offset to the nearest ground-truth box [56]. At last, the top predictions from these two subnetwork-layers are merged followed by non-maximum suppression with a specific threshold value (0.5) to yield the final bounding boxes with their corresponding class as well as the probability value [56].

2.2.9 Detection Transformer (DETR)

Detection Transformer (DETR) was designed to see the potential of object detection as a regression problem instead of using other techniques, *e.g.* non-maximum suppression or anchor generation as in RPN. DETR uses a technique built on an Encoder-Decoder architecture based on a recently developed method named Transformers [58]. The overall architecture of this model is mainly built on three different components, more specifically a CNN backbone, the Encoder-Decoder Transformer block mentioned earlier and lastly a feed-forward network. This is visualised in Figure 2.20.

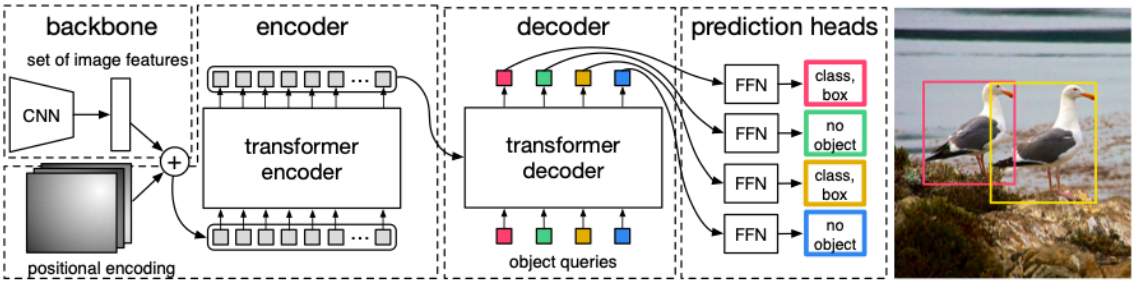


Figure 2.20: The architecture of the DETR model which is represented by three different layers, more specifically the CNN backbone, the Encoder-Decoder Transformer block and a feed-forward network (FFN). From [59].

The first stage is quite similar to previous models, more specifically a CNN backbone that creates a convolutional feature map from an image. Uniquely for DETR is that the feature map is transformed to a one-dimensional feature map, since the following layer in the model expects a single array as input [58]. Then, the Transformer encoder uses the one-dimensional feature vector as an input [58]. The Transformer encoder is built of various layers, and each layer is mainly built on two sub-layers, more specifically a multi-head self-attention mechanism and a fully-connected feed-forward network (FFN) [60]. The multi-head self-attention layer allows the entire model to attend information from different representation subspaces at various locations [60]. The output from this layer is a specific number of fixed length vectors where the number of vectors are equivalent to the number of objects in the image assumed by the system [58].

The next step in the model is the Transformer decoder with similar design as the previous layer [58]. By using self-attention and Encoder-Decoder attention mechanisms, the decoder uses the convolutional feature vectors and estimates bounding box coordinates [58]. At last, those bounding box coordinates are translated to normalised center coordinates as well as height- and width values of the bounding box with respect to the input image [58]. A FFN-block follows the Transformer decoder mainly made of a linear classifier named as a 3-layer perception including some ReLU-functions as well as a linear projection layer [58]. Its purpose is to predict the results in form of normalized center coordinates of the bounding box and its corresponding height and width values. This is done at the same time as the linear projection layer (*e.g.* a softmax layer) predicts the corresponding class to the

bounding box [58]. This system predicts much more labels in the feature vectors compared to the interesting number of labels, which leads to that a specific class named $\varnothing$ is introduced [58]. This results in that the background class of an images is represented by the following label, $\varnothing$, which means no object.

The second unique thing with the DETR model is the loss function used to be minimised in order to get accurate results. The name of the loss function is the Hungarian loss, where the first term in the sum represents the loss from the class prediction and the second term represents the box loss [58]. The Hungarian loss is visualised below:

$$L_{Hungarian}(y, \hat{y}) = \sum_{i=1}^{N} [\ -\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbf{1}_{\{c_i \neq \varnothing\}} L_{box}(b_i, \hat{b}_{\hat{\sigma}}(i))]\ , \tag{2.19}$$

where y is the ground truth set of objects, $\hat{y}$ is the set of N predictions, c_i is the target class label, b_i is the ground truth bounding box center coordinates vector, $\hat{p}_{\sigma(i)}(c_i)$ is the prediction, $\hat{b}_{\sigma(i)}$ is the predicted box coordinates and L_{box} is the bounding box loss [58]. DETR matches the ground truth set of objects with the set of predictions [58]. This is done by calculating the lowest cost using the following expression:

$$\hat{\sigma} = \underset{\sigma \in \mathfrak{S}_N}{\arg\min} \sum_{i}^{N} L_{match}(y_i, \hat{y}_{\sigma(i)}), \tag{2.20}$$

where L_{match} is a pair-wise matching cost [58]. After the pairs have been matched, the Hungarian loss can be calculated. In order to calculate the pair-wise matching cost, the expression below can be used:

$$L_{match}(y_i, \hat{y}_{\sigma(i)}) = -\mathbf{1}_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}(c_i) + \mathbf{1}_{\{c_i \neq \varnothing\}} L_{box}(b_i, \hat{b}_{\sigma(i)}). \tag{2.21}$$

$\hat{\sigma}$ is the optimal assignment calculated using Equation (2.20) [58]. In order to estimate the Hungarian loss, the bounding box loss needs to be calculated. This can be done as follows:

$$L_{box}(b_i, \hat{b}_{\sigma(i)}) = \lambda_{iou} L_{iou}(b_i, \hat{b}_{\sigma(i)}) + \lambda_{L1} ||b_i - \hat{b}_{\sigma(i)}||_1, \tag{2.22}$$

where λ_{iou} and λ_{L1} are real-valued hyperparameters used to control the learning process, and L_{iou} is the generalized IoU loss [58]. The generalized IoU loss is quite simple to estimate by using the following expression:

$$L_{iou} = 1 - IoU. \tag{2.23}$$

In DETR, ResNet is often used as a backbone of the model which in turn uses connections similar to skip connections [58].

2.2.10 VarifocalNet (VFNet)

VarifocalNet (VFNet) is an object detection model with mainly three unique head-steps in the design. More specifically, an IoU-aware Classification Score (IACS) that is learned as a joint representation of object presence and localization accuracy in combination with a designed loss function (Varifocal loss) to train a dense object detector to predict the IACS values [61]. This is combined with a newly designed star-shaped bounding box, with a refinement loop based on the IACS predictions in order to improve the predictions of the bounding boxes [61]. The IoU-based detector is inspired by two different techniques, more specifically a fully convolutional one-stage object detector (FCOS) that does not uses pre-defined anchor boxes such as RetinaNET, Faster R-CNN and YOLOv3, and also an Adaptive Training Sampling Selection (ATSS) algorithm in order to select positive and negative samples based on statistically characteristics of objects [61]. FCOS and ATSS are not be described more thoroughly in this report. The architecture of the VFNet is represented in Figure 2.21.

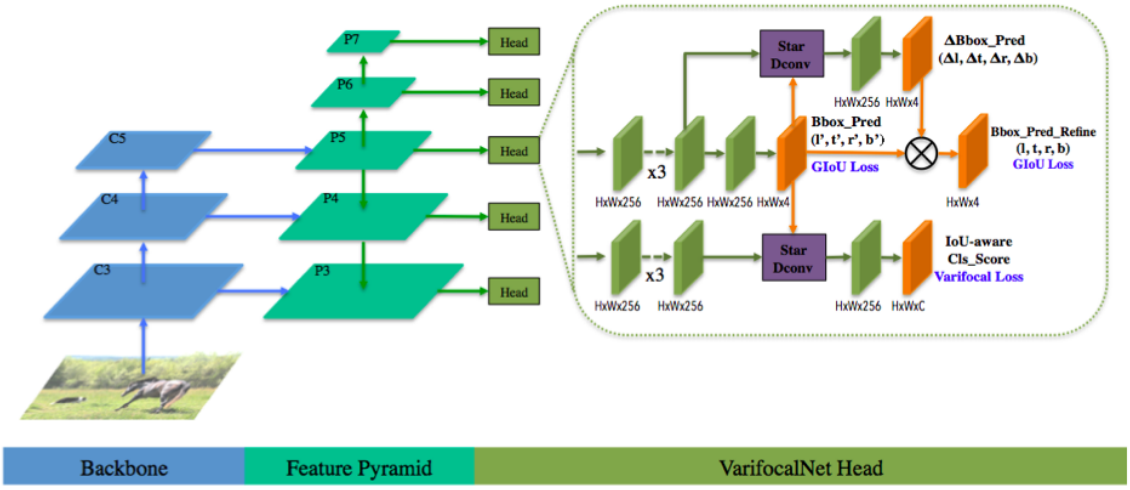


Figure 2.21: The network structure of VFNet, consisting of its backbone connected via feature pyramids to the VarifocalNet head. From [62].

As mentioned earlier, VFNet has three unique techniques in its head, which is the most significant part of the model. First of all, IACS is used to rank the detections, and is structured as a classification score vector with the length equal to the number of classes [61]. Each class is represented with a value. For the ground truth class, the value is the IoU over the predicted and the ground truth bounding box, and for the other classes predicted with zero values [61].

To predict the IACS, VFNet uses the Varifocal loss presented as follows:

$$VFL(p,q) = \begin{cases} -q(q\ log(p) + (1-q)log(1-p))\ q > 0 \\ -\alpha p^{\theta} log(1-p)\ q = 0, \end{cases} \tag{2.24}$$

which is a loss function inspired by the focal loss and the binary cross entropy loss [61]. In Equation (2.24), p is the predicted IACS-vector, q is the target score mean-

ing that $q = 0$ for the background and for the foreground it is described by the IoU-value of the predicted and ground truth bounding boxes in the ground truth class. As mentioned earlier, the rest of the classes are described with zero values [61]. The variable α is an adjustable scaling factor to balance the loss between the negative and positive examples [61]. The loss function decreases the contribution from the negative examples and preserves the effect on the loss from the positive examples. The positive examples are more rare and therefore give a larger contribution [61].

Next part of VFNet is the star-shaped bounding box feature representation. VFNet uses a bounding box with deformable convolution, containing nine fixed sampling points [61]. The nine points:

$$\text{sampling point} = \begin{bmatrix} x, y \\ y - l', y + b' \\ x, y - t' \\ x + r', y \\ x, y + b' \\ x - l', y - t' \\ x + l', y - t' \\ x - l', y + b' \\ x + r', y + b', \end{bmatrix} \tag{2.25}$$

are created using a four dimensional vector (l', t', r', b') that is the distances from the point (x, y) in the directions: left, top, right and bottom [61]. From the sampling point in Equation (2.25), the relative offset to the projected point (x, y) is used in the deformable convolution, meaning that the features at these nine positions are convolved to give the bounding box [61]. VFNet uses deformable convolution, which is a method that in contrast to normal convolution with a fixed rectangular kernel uses a kernel with an offset, making the convolution method more flexible [63]. Deformable convolution can be described as follows:

$$y(p_0) = \sum_{p_n \in R} w(p_n) * x(p_0 + p_n + \Delta p_n). \tag{2.26}$$

In Equation (2.26), $y(p_0)$ is the calculated output value from the kernel in the point p_0, p_n is a point in R and R are the positions of the kernel, w is the kernel weights and lastly Δp_n is the offset of the point p_n [63]. The bounding box is changed by learning during the bounding box refinement, which is done using regression [61]. The distances from the prediction is calculated as scaling factors $(\Delta l, \Delta t, \Delta r, \Delta b)$ and presented as follows:

$$(l, t, r, b) = (\Delta l \times l', \Delta t \times t', \Delta r \times r', \Delta b \times b'). \tag{2.27}$$

This is done to refine the prediction [61].

2.3 Data Augmentation

In order to increase the amount of training data, different techniques can be used. Data augmentation is a common name for these techniques used in data analysis, *e.g.* computer vision tasks. This results in more various types of the images and in turn a larger dataset, which often leads to better performance of a model [64]. Additionally, an augmented set of data can reduce the overfitting problem that can occur if the dataset is small, or if the images are quite similar to each other [64]. There are different techniques used in this area to get unique data from existing images. In the following sections, some common data augmentation techniques are presented.

2.3.1 Resizing

Resizing is a data augmentation technique that is quite simple. The input images are cropped or padded to a specific size, which leads to an increased number of images in the dataset [64].

2.3.2 Random Flip

Random flip is a quite common data augmentation technique that either flip an image vertically or horizontally. This technique is used in order to increase the robustness of the system [65].

2.3.3 Rotation

Rotation is also a specific technique that increases the robustness of the system by rotating the image and its corresponding bounding boxes by a specific angle [65]. This can be done in several ways. Either, the image is rotated by a predefined angle, or the angle is randomly defined before the rotations [65]. Another implementation often used is to randomly select an angle that is redefined after each rotation. This can lead to a diversified dataset [65]. A rotated point defined by its x and y-coordinates $((x_r, y_r))$ are rotated by multiplication of the original coordinated $((x, y))$ with the rotation matrix R below [66]:

$$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{2.28}$$

2.3.4 Using Kernel Filters

Kernel filters are popular techniques used in data augmentation purposes in order to change the quality of the image, which often also increase the robustness and the variety of the data [65]. The variation of the data can be created in several ways. One implementation is to blur the image by using Gaussian Blur filters, while another example is to use image sharpening in order to highlight details in the images [65].

2.3.4.1 Gaussian Blur Filter

Gaussian Blur filters are often used to smooth edges in the image which describes the results of taking a photo with a de-focused lens [67]. Since this type of filter is used to smooth the edges in an image, it belongs to the group of low-pass filters [67]. A Gaussian Blur filter is expressed by the normal distribution in two directions, and can be seen below:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{2.29}$$

where x and y is the distance of the kernel in each direction and σ is the standard deviation [67].

2.3.4.2 Image Sharpening Filter

Image sharpening filters are a group of filters within high-pass filters, which highlight details in an image and de-emphasize areas in the image with slowly varying intensities [67]. This technique is quite common in medical purposes, *e.g.* in X-ray imaging [67]. There are several sharpening filters available, but one common method is to subtract a Laplacian filtered image from the original image in order to highlight details [67]. Often, this can be done before the implementation of the kernel by subtracting the identity matrix with the Laplacian kernel. The sharpened kernel is presented as:

$$\text{Sharpened Kernel} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \tag{2.30}$$

and the kernel is applied by convolution with the input image [67].

2.3.5 Changing Parameters in Color Space

Another common method to improve the robustness of the system is to make augmentation in the color channel space. This results in different changes of the input image. Either, the color of the image can be changed by changing the hue or via changing the saturation and/or the brightness [65]. All of these techniques results in relatively new images, since color is an important parameter in image representation.

2.4 Evaluation Method - mean Average Precision

Evaluation of object detection methods needs to consider both detection and localization performance, which requires a comprehensive metric. It is common in object detection purposes to use Average Precision (AP) and mean Average Precision (mAP). The mAP is based on the IoU over several specific threshold values. The threshold value is used to distinguish between positive predictions and negative predictions, since the IoU estimates the ratio of the overlap between the ground truth and the predicted bounding box. In order to estimate the AP, two new essential parameters need to be introduced. More precisely, these two parameters are the

Precision and Recall values. Precision is defined as the accuracy of the predictions, while Recall is defined as the percentage of how many positive examples the model can find. Those two parameters can be seen below:

$$\text{Precision} = \frac{TP}{(TP + FP)}, \tag{2.31}$$

and

$$\text{Recall} = \frac{TP}{(TP + FN)}, \tag{2.32}$$

where TP represents the True Positive examples, FP represents the False Positive examples and FN represents the False Negative examples. TP are bounding boxes that are correctly classified, FP are the bounding boxes that are classified as foreground but they are background in the ground truth, and FN are the boxes that are classified as background but belong to foreground in the ground truth. The Precision and Recall values can be plotted against each other with Precision at the y-axis and Recall at the x-axis. The integral under the curve defines the AP score which is in range between 0 and 1, since the Precision and Recall properties are bounded by the same values. AP is a popular measure in object detection to calculate the accuracy of the model.

From the AP value, mAP can be estimated. There are different definitions of the mAP depending of the purpose. We have chosen to use the COCO-method in order to estimate the mAP. Often, mAP is estimated as an average of 10 different AP-values from certain IoU-thresholds. The lowest threshold value used is 0.5 and the highest is 0.95 with a step size equal to 0.05. The mAP is presented for each class and also commonly displayed with an averaged value over all classes. The calculations are expressed as follows:

$$\text{mAP@}[0.5 : 0.95] = \frac{1}{10} \sum_{i \in I} AP_i \,\text{where}\, I = [50, 55, 60, 65, 70, 75, 80, 85, 90, 95]. \tag{2.33}$$

3
Methodology

This project is divided into two parts, the creation of the dataset and the testing of different models for this application. At first, this chapter describes the procedure behind the creation of the dataset from the colonoscopy videos, and then the methodology behind the investigation of existing object detection models. Finally, the new dataset is used in combination with the best performing models to evaluate the potential of the application.

3.1 Creation of the Dataset

The creation of the dataset has been done in a specific program named Augere Video Tagging, which is a labeling program used in previous similar studies. A total of 20 videos, consisted of RGB-frames with a size equal to 1024×576 pixels, were investigated through the tagging tool program in collaboration with a medical student.

The videos were recorded during real colonoscopy investigations from patients in Norway diagnosed with ulcerative colitis. When uploading the videos to the annotation tool, they were anonymised because of their sensitive information. The labeling was done with supervision from medical specialists. The videos were reviewed and biopsies and polyp removal were annotated at first. An important step was also to label obscure objects in the same frames as pre-labelled objects. Otherwise, the network is not be trained in the best way since unlabelled findings are present in the background. Thomas de Lange, in collaboration with other experts in the area, examined the objects labelled and changed to the correct diagnose after discussions.

The annotated frames with corresponding labels were exported to be used in the dataset, and examples can be seen in Appendix A.1:

- Hyperplasia (Biopsy)
- Hyperplasia (Polyp)
- Dysplasia (Biopsy)
- Dysplasia (Polyp)
- Inflammation (Biopsy)
- Inflammation (Polyp)

To minimize the number of equal images when extracting several frames of the same object, a selection of every 5th frame was done, resulting in a total of 840 images. To improve the results of the tests in lack of material, the two classes of hyperplasia

was merged into one. The same was done with the two labels of dysplasia.

3.2 Investigation of Existing Object Detection Models

Since the new dataset was created in parallel to the investigation of existing object detection models, a previously developed dataset, Kvasir, was used as a pre-study while waiting for the finishing of the new dataset. Kvasir is made of medical images from the GI-tract [1]. The images are taken related to endoscopic polyp removal and divided into different anatomical landmarks and pathological findings [1].

Mainly three different repositories available on GitHub were investigated and used in combination with Google Colab. The first repository named Darknet developed by AlexeyAB hosts different versions of the object detection model YOLO [68]. Secondly, another repository named MMDetection developed by OpenMMLab where different networks and methods are available to use [69]. Lastly, the third repository that has been investigated is the Tensorflow Object Detection available on GitHub [70]. The Tensorflow's repository contains various selections of models as well [70].

A method was chosen on their corresponding repository and implemented according to the instructions by downloading their pre-trained weights. The pre-trained weights are often trained on a large dataset (*e.g.* the COCO-dataset, which is a dataset with more than 200000 labeled images with 80 object categories [71]). This helps the system to be faster in the training procedure through transfer learning.

The Kvasir dataset was split into 800 training images, 100 validation images and finally 100 test images. The fractioning of the images in the different sets were done randomly. This was done to minimizes the bias term that can occur if the user has a specific intention with the split of the dataset. As described in the theory, mAP was used as a validation metric.

3.3 Data Augmentation

The augmentation is done in different ways depending of the model. There are some augmentation techniques that are used on-the-fly, while other techniques are implemented before the training procedures which results in a larger dataset from start. The models used from MMDetection repository use the techniques resizing and random flip on-the-fly. The YOLO-models used from AlexeyAB Darknet repository use color changing parameters on-the-fly, more specifically randomly changing brightness, saturation and hue. The last repository used named Tensorflow Object Detection uses random images cropping, random flip and padding the images to squares. Those techniques are also used on-the-fly.

There are three different techniques investigated in order to create a larger dataset. Different ratios of these techniques were also investigated. More specifically, 25%, 50% and 75% of the original training dataset was augmented and added when using Kvasir. For the new dataset, all augmentation techniques were used in combination. Firstly 25/50% of the dataset were rotated with random angles and the rotated images were added to the dataset. After the addition of the rotated images, 25/50% of the images were blurred and 25/50% were sharped and added to create the augmented dataset, which is illustrated in Figure 3.1.

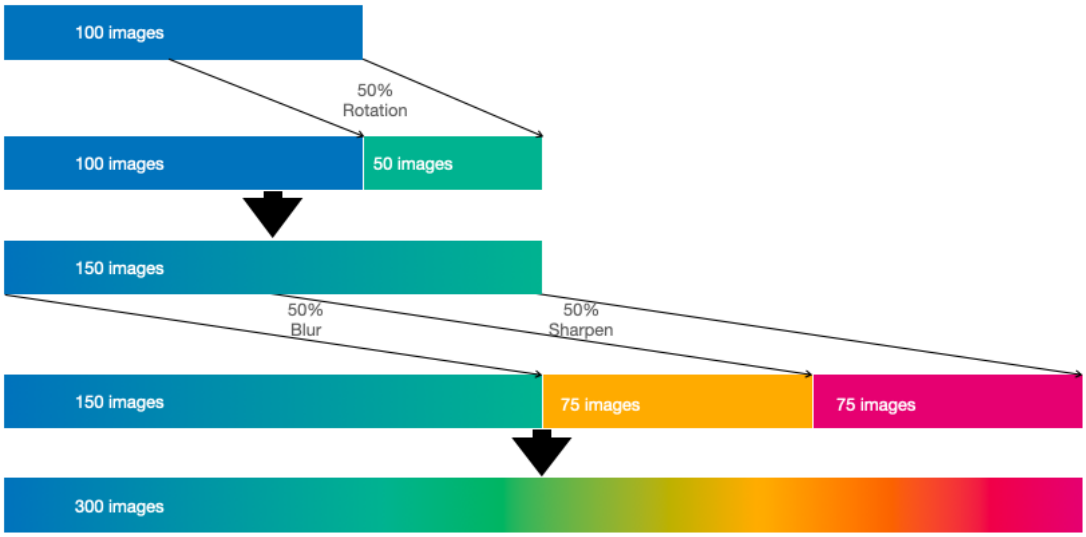
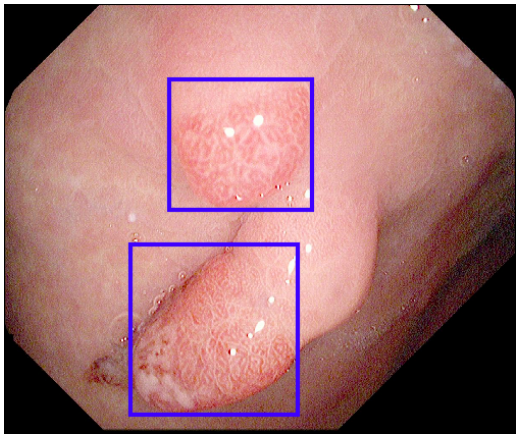


Figure 3.1: The new dataset were augmented so that firstly 25/50% of the dataset were rotated with random angles and after the addition of the rotated angles, 25/50% of the images were blurred and 25/50% were sharped.

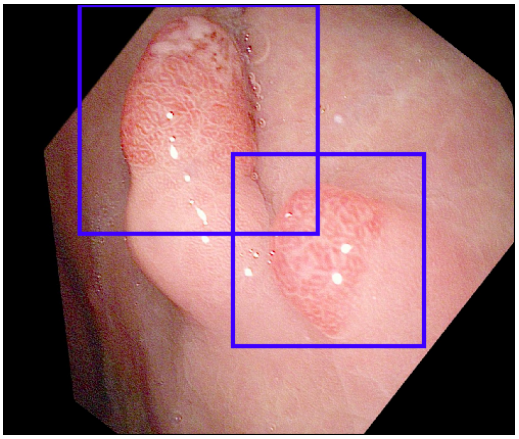
The choice of using 25% and 50% was due to computational expenses and the results of the ablation study. Since blurring and sharpening are each others' antagonists, this arrangement was done to minimize the risk of getting images processed with both these techniques.

3.3.1 Batch Rotation

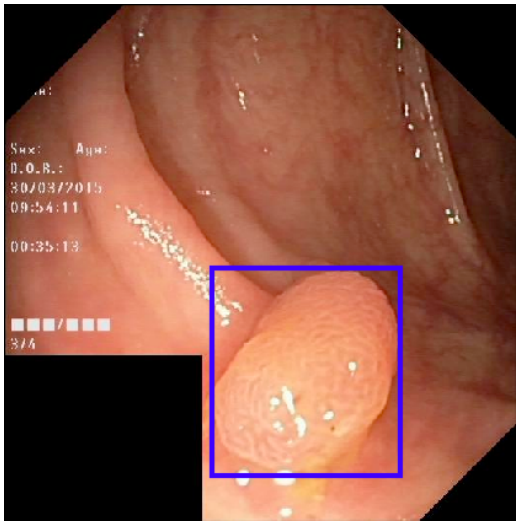
In some of the experiments, rotation was used in order to create more images. The images were rotated by a specific angle chosen randomly within a range from 0 to 365 degrees. Described in the previous section, three different ratios were used. Some examples of rotated images and their corresponding bounding boxes can be seen in Figure 3.2, where the original images from Kvasir are visualised in Figures 3.2a and 3.2c. The rotated images are presented in Figures 3.2b and 3.2d.



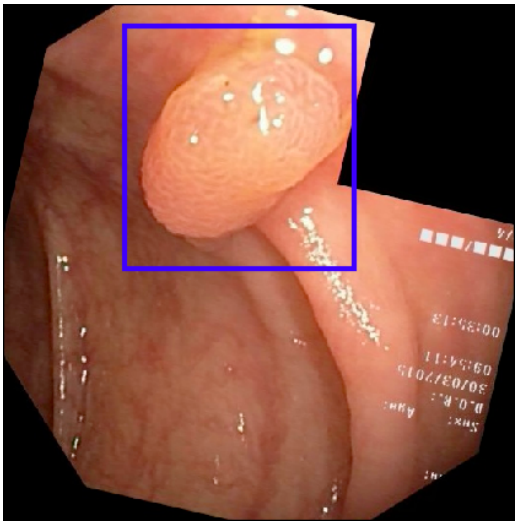
(a) Image from Kvasir and its corresponding bounding boxes.



(b) Rotated image and its corresponding bounding boxes.



(c) Image from Kvasir and its corresponding bounding box.



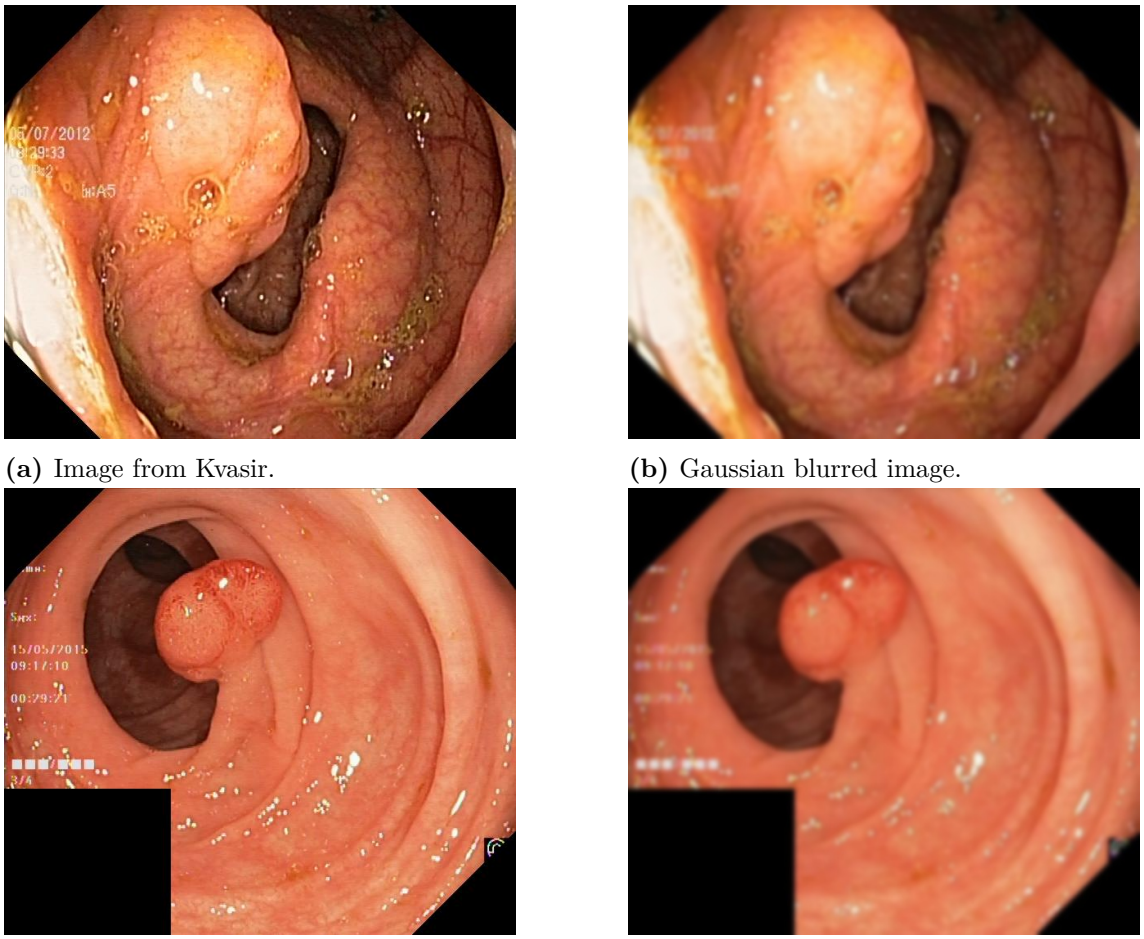
(d) Rotated image and its corresponding bounding box.

Figure 3.2: The result of rotating some images from the dataset Kvasir with a specific angle chosen randomly. From [72].

Note that the bounding boxes visualised in 3.2b and 3.2d are larger compared to the original bounding boxes presented in 3.2a and 3.2c, which is an undesired but inevitable result from the rotation matrix.

3.3.2 Gaussian Blur Filter

Different kernel sizes and standard deviations were tested in order to get appropriate blurred images. After some testing, the kernel size equal to 21 pixels in both dimensions and the standard deviation (σ) equal to 3 were chosen. In Figure 3.3, two examples of the images in Kvasir (3.3a and 3.3c) with their corresponding blurred images (3.3b and 3.3d) can be seen.



(a) Image from Kvasir.

(b) Gaussian blurred image.

(c) Image from Kvasir.

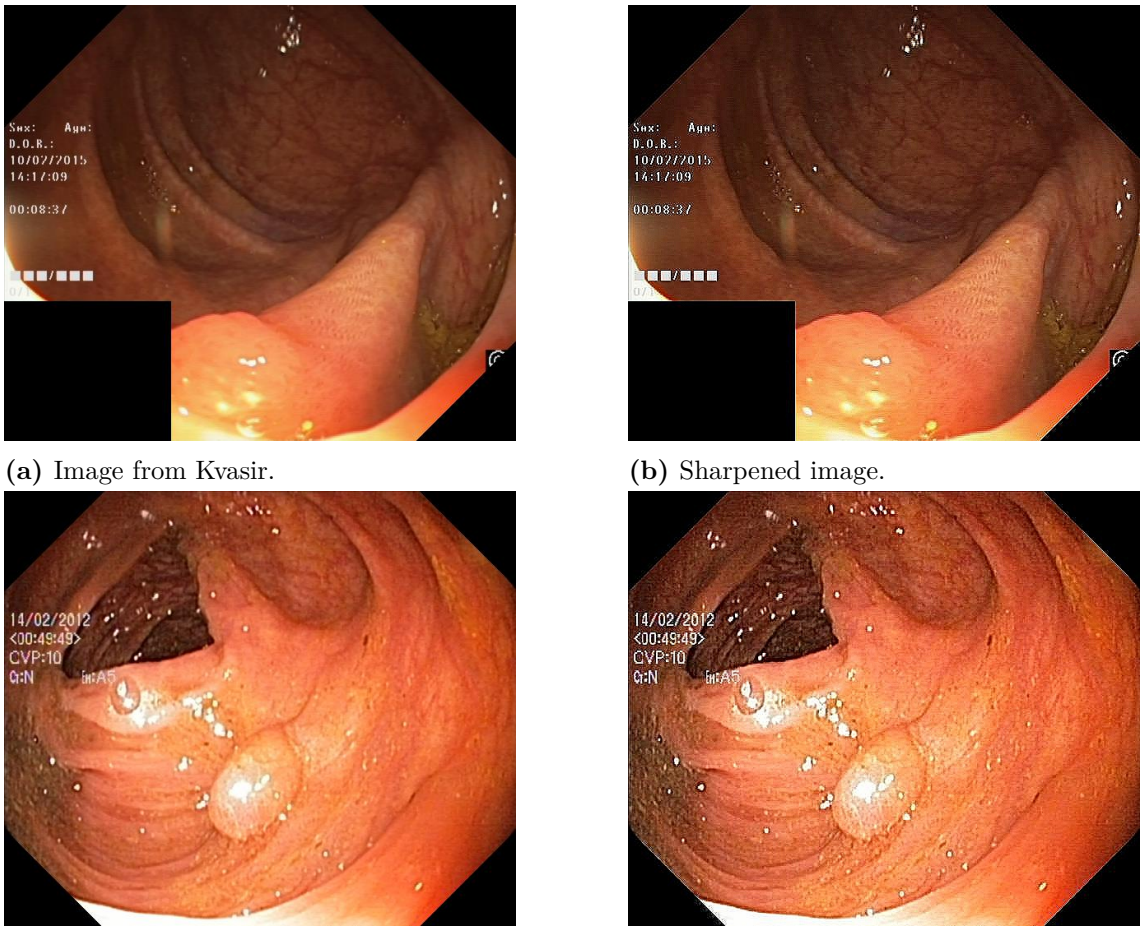
(d) Gaussian blurred image.

Figure 3.3: The result of blurring two images from the dataset Kvasir using Gaussian Blur filter with kernel size equal to [21, 21] pixels and standard deviation equal to 3. From [72].

From the images in Figure 3.3, it can clearly be seen that the images have been blurred which can be compared with a de-focused lens from a camera.

3.3.3 Image Sharpening

The sharpened kernel expressed in Equation (2.30) was used in convolution with the input image to create sharpened images. In Figure 3.4, two examples of images from Kvasir (3.4a and 3.4c) and their corresponding sharpened images (3.4b and 3.4d) are presented.



(a) Image from Kvasir.

(b) Sharpened image.

(c) Image from Kvasir.

(d) Sharpened image.

Figure 3.4: The result of sharpen two images from the dataset Kvasir using the image sharpening filter expressed in Equation (2.30). From [72].

By analysing the images in Figure 3.4, it can be seen that the details in the images are highlighted by the sharpening filter, *e.g.* detailed blood vessels in Figure 3.4b.

3.4 Comparison with a medical expert

In order to see if there are visual macroscopic differences in the frames from the new dataset that can be seen by a trained eye, a smaller test was done. 50 frames from the new dataset were selected randomly and presented to a medical student for classification with the three classes, more specifically dysplasia, hyperplasia and inflammation. The medical student labelled the frames by its macroscopic investigation, and the results were compared with the correct microscopic results. The annotation was done with only labels, which means that the localisation task was removed from the purpose. This was done because the most important aspect was to see if the medical student could distinguish between those classes.

4
Results

In this section, the results from the dataset and the tests are shown. The tests are done on both the Kvasir dataset and the new dataset, and the results are presented in terms of mAP. Lastly, a small section about bench-marking the performances of the models are presented.

4.1 Creation of the New Dataset

From 10 different colonoscopy videos, the findings were labeled and every fifth frame from each finding was extracted. Then, the first dataset was created, where the actual numbers for each class are presented in Table 4.1. The training and validation split was made depending on the associated videos in order to get rid of the bias term that can occur if similar frames were placed in both the training- and the validation set.

Table 4.1: Numbers of frames for each class, where every fifth frame was extracted from the colonoscopy videos.

Class	Frames
Dysplasia	252
Hyperplasia	196
Inflammation (Polyp)	166
Inflammation (Biopsy)	226

Later in the process, a dataset was created that only took into account frames with classes either equal to dysplasia (polyp), hyperplasia (polyp) or inflammation (polyp). Those three classes was later described with the following names: dysplasia, hyperplasia and inflammation respectively. In Table 4.2, the actual numbers of frames for the narrowed dataset are presented.

Table 4.2: Numbers of frames for each class that contains the name "(Polyp)", where every fifth frame was extracted from the colonoscopy videos.

Class	Frames
Dysplasia	175
Hyperplasia	136
Inflammation	166

4.2 Kvasir Dataset

At first, a study was done using the Kvasir dataset made of 1000 colonoscopy videos. As described earlier in the methodology, the dataset contained bounding boxes and one class, more specifically Polyp. Ten different models were trained with pre-trained weights available on the corresponding GitHub repository. The results from those models in form of the validation accuracy represented with mAP scores can be seen in Table 4.3.

Table 4.3: The mAP scores for the different models investigated.

Model	mAP@[0.5 : 0.95]
Faster R-CNN	72.9
RetinaNET	76.3
EfficientNET_d0 (512x512)	59.1
EfficientNET_d2 (768x768)	65.6
YOLOv3 (416x416)	63.0
YOLOv3 (608x608)	46.7
YOLOv4 (416x416)	69.1
YOLOv4 (608x608)	60.8
DETR	57.9
VFNet	**79.9**

As can be seen in Table 4.3, the majority of the models achieve a MAP score higher than 60% on the validation set. From those seven models, only three have a higher accuracy than 70% of the mAP value. More specifically, the best performing models on the validation set are Faster R-CNN, RetinaNET and VFNet, which achieved 72.9%, 76.3% and 79.9% respectively. The training of the various models was stopped when the mAP-values for the different epochs started to converge. An example of a mAP-graph plotted against the actual number of epochs can be seen in Figure 4.1, where the training in this case was stopped at the sixth epoch.

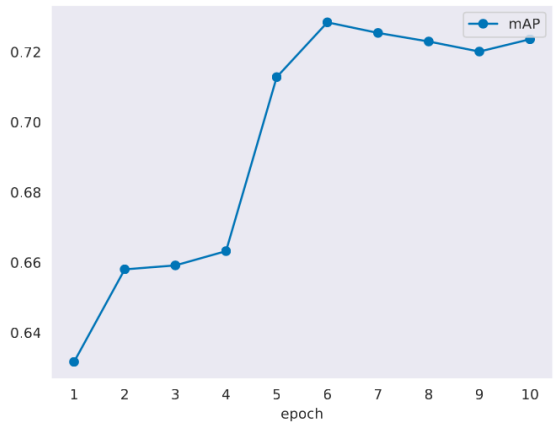
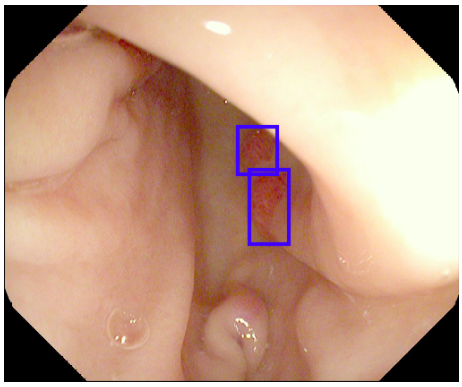


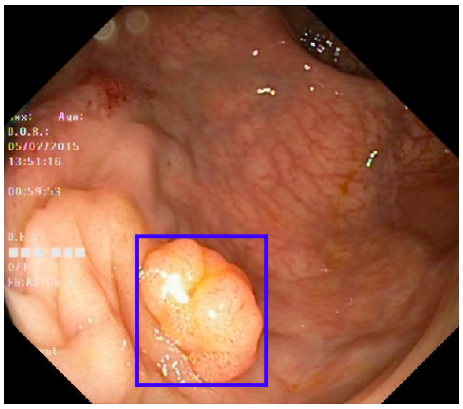
Figure 4.1: A mAP-graph plotted against the number of epochs for Faster R-CNN.

The models trained on the Kvasir dataset were used on the test set in order to

visualise the results. In Figure 4.2, two negative examples are presented where the models have predicted bounding boxes with quite low precision. In Figure 4.2c and 4.2d, DETR predicted folded mucosa as a polyp in the first image, and folded mucosa in the second image was wrongly predicted. As can be seen in Figure 4.2e and 4.2f, Faster R-CNN predicted two bounding boxes wrongly in the first image, and had the same problem with folded mucosa as DETR. The ground truth images are visualised in Figure 4.2a and 4.2b.

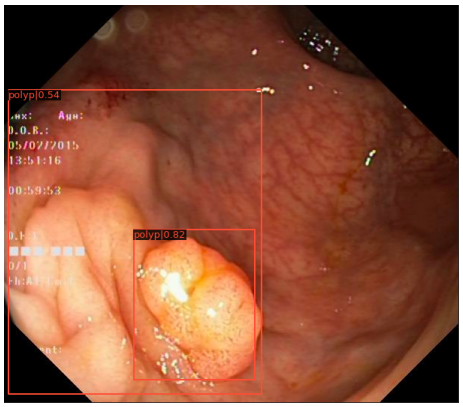


(a) Image from Kvasir and its corresponding ground truth bounding boxes.

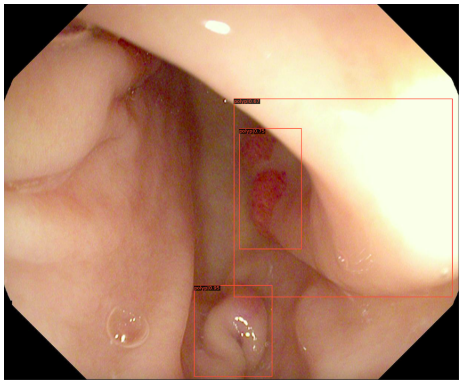


(b) Image from Kvasir and its corresponding ground truth bounding box.

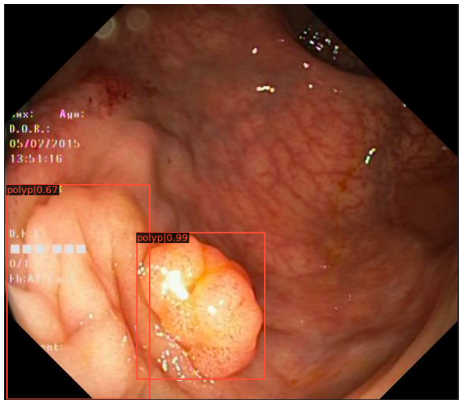
(c) Result of using DETR.



(d) Result of using DETR.



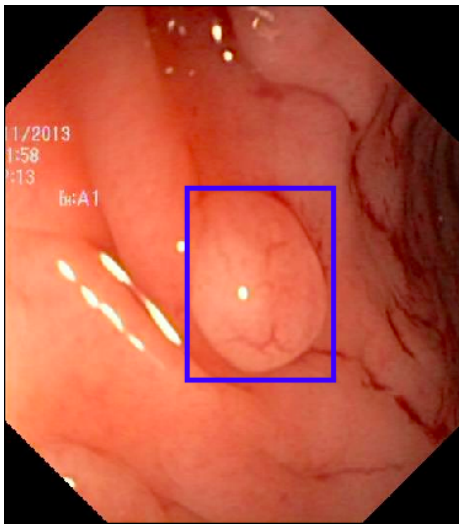
(e) Result of using Faster R-CNN.



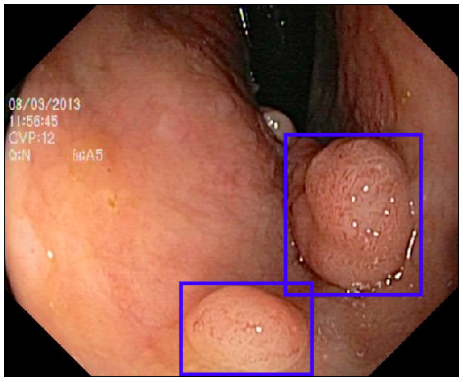
(f) Result of using Faster R-CNN.

Figure 4.2: Examples showing the performance of DETR and Faster R-CNN operating on the Kvasir dataset, where the detections are done with low precision. From [72].

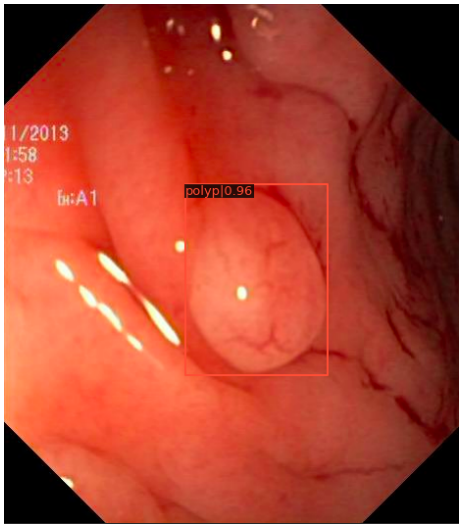
Since the majority of the models achieved high accuracy on the validation set, they also performed well on labeling and predicting objects in the test set. Examples can be seen in Figure 4.3, where VFNet (Figure 4.3c and 4.3d) as well as Faster R-CNN (4.3e and 4.3f) predicted bounding boxes and labels with high precision. Note also in the same images that the models can handle multiple objects in the same images.



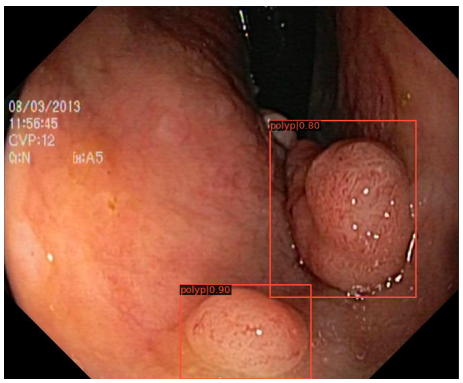
(a) Image from Kvasir and its corresponding ground truth bounding boxes.



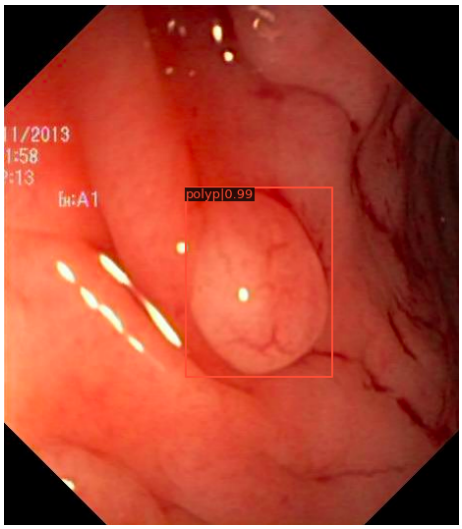
(b) Image from Kvasir and its corresponding ground truth bounding box.



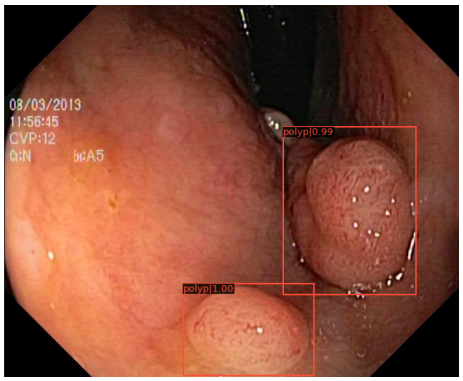
(c) Result of using VFNet.



(d) Result of using VFNet.



(e) Result of using Faster R-CNN.



(f) Result of using Faster R-CNN.

Figure 4.3: Examples showing the performance of Faster R-CNN and VFNet operating on the Kvasir dataset, where the detections are done with high precision. From [72].

For further investigations, the three best performing models were selected. More specifically, these models were Faster R-CNN, RetinaNET and VFNet.

4.2.1 Batch Rotation

Batch rotation was done on the training set in three different ratios, 25% 50% and 75% and tested on Faster R-CNN, RetinaNET and VFNet. The results for the three models when 25% of the training set was augmented using rotation and added to the existing set can be seen in Table 4.4.

Table 4.4: The mAP scores for the validation set when 25% of rotated images were added to the training set.

Model	mAP@[0.5 : 0.95]
Faster R-CNN	71.1
RetinaNET	74.2
VFNet	77.7

By comparing the values presented in Table 4.4 with the values in Table 4.3, it can be seen that the augmented dataset achieved lower accuracy than the original set. The augmented ratio was then increased to 50%, and the results can be seen in Table 4.5. By analysing the results from that experiment, it can be seen that the mAP scores are lower for each model compared to the values presented in Table 4.4.

Table 4.5: The mAP values for the validation set when 50% of rotated images were added to the training set.

Model	mAP@[0.5 : 0.95]
Faster R-CNN	68.9
RetinaNET	73.6
VFNet	74.2

In Table 4.5 it can be seen that the mAP score increases for VFNet but not for the other models in comparison to 25% rotation in Table 4.4. At last, 75% of the training dataset was rotated and added to the original set, which can be seen in Table 4.6.

Table 4.6: The mAP scores for the validation set when 75% of rotated images were added to the training set.

Model	mAP@[0.5 : 0.95]
Faster R-CNN	71.4
RetinaNET	74.7
VFNet	80.6

From those results, it can be seen that there is slightly an improvement from VFNet compared to the value presented in Table 4.3, while the other values for Faster R-

CNN and RetinaNET are slightly lower. Note that VFNet is the best performing model for each augmented rotation ratio.

4.2.2 Gaussian Blur Filter

Similar to the batch rotation experiments, the original training set was augmented using a Gaussian Blur filter with similar ratios, more specifically 25%, 50% and 75%. The results from the experiments where 25% of the training set was augmented can be seen in Table 4.7. By comparing those results with Table 4.3, it can be seen that Faster R-CNN and VFNet performs quite the same as before the augmentation, while RetinaNET is around 10 percentage points lower in accuracy than before this augmentation technique was used.

Table 4.7: The mAP values for the validation set when 25% of Gaussian Blurred images were added to the training set.

Model	mAP@[0.5 : 0.95]
Faster R-CNN	69.2
RetinaNET	67.3
VFNet	77.5

When 50% of the training set was augmented with blurred images, it can be seen that the accuracy is quite the same for VFNet, while there is a slightly improvement in mAP for Faster R-CNN and RetinaNET compared to Table 4.3. The results are presented in Table 4.8.

Table 4.8: The mAP scores for the validation set when 50% of Gaussian Blurred images were added to the training set.

Model	mAP@[0.5 : 0.95]
Faster R-CNN	75.9
RetinaNET	78.5
VFNet	80.1

Lastly, 75% of the training set was augmented and added, which led to quite the same results as in Table 4.3 for Faster R-CNN, while the mAP values for RetinaNET and VFNet are slightly lower. The results from this experiment are presented in Table 4.9.

Table 4.9: The mAP values for the validation set when 75% of Gaussian Blurred images were added to the training set.

Model	mAP@[0.5 : 0.95]
Faster R-CNN	72.8
RetinaNET	73.9
VFNet	77.2

4.2.3 Image Sharpening

Even in this case, the same ratios of augmentation was used with a new augmentation technique, more specifically adding sharpened images using a sharpening filter. In Table 4.10, the mAP-values are presented when 25% of the images was sharpened and added to the training set. Note that the accuracy for each model is about 10 percentage points lower compared to the results in Table 4.3.

Table 4.10: The mAP values for the validation set when 25% of sharpened images were added to the training set.

Model	mAP@[0.5 : 0.95]
Faster R-CNN	62.8
RetinaNET	66.3
VFNet	70.5

When 50% of the training set was augmented with this technique, it can be seen that the accuracy in Table 4.11 is quite the same as the values in Table 4.3. Note also that 50% augmented training set gives higher accuracy for each model compared with the results in Table 4.10 where 25% was augmented.

Table 4.11: The mAP values for the validation set when 50% of sharpened images were added to the training set.

Model	mAP@[0.5 : 0.95]
Faster R-CNN	70.7
RetinaNET	76.3
VFNet	79.5

The last experimental design in this case was to augment 75% of the training set with sharpened images. These results can be seen in Table 4.12.

Table 4.12: The mAP values for the validation set when 75% of sharpened images were added to the training set.

Model	mAP@[0.5 : 0.95]
Faster R-CNN	63.3
RetinaNET	66.2
VFNet	71.2

In this case, the accuracy in Table 4.12 for each model can be compared to the values in Table 4.10 which have similar results. From this section, it can be seen that augment the training set with 50% sharpened images gave the highest accuracy.

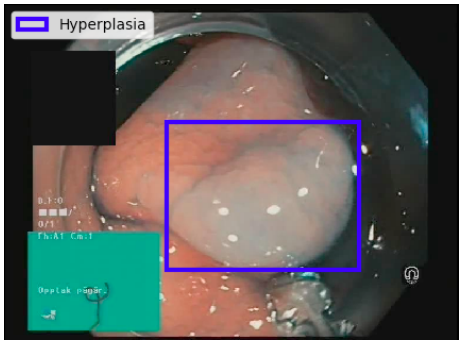
4.3 New Dataset

The three best performing models on the dataset Kvasir, more specifically Faster R-CNN, RetinaNET and VFNet, were used in combination with the newly created dataset. To begin with, three training procedures were started with the same pre-trained weights as in the previous case with Kvasir. The only augmentation techniques used in this stage are the ones on-the-fly. The results from using Faster R-CNN as the first model are visualised in Table 4.13.

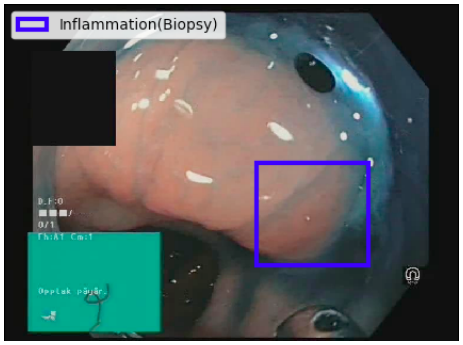
Table 4.13: The mAP values for the different classes using the model Faster R-CNN are presented. The AP-values for 0.5 and 0.75 are also visualised.

Class	AP_{50}	AP_{75}	mAP@[0.5 : 0.95]
Dysplasia	0	0	0
Hyperplasia	34.2	11.6	13.7
Inflammation (Polyp)	4.5	0.1	1.4
Inflammation (Biopsy)	0	0	0
All classes			3.8

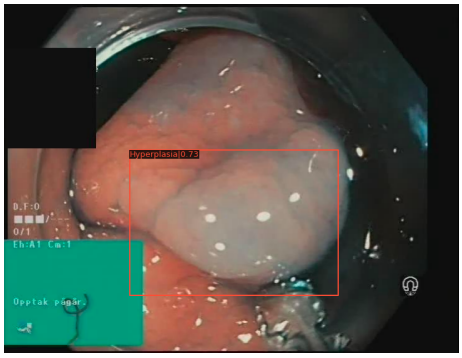
As can be seen in Table 4.13, Faster R-CNN cannot detect dysplasia or inflammation (Biopsy), and the rest of the classes have quite low accuracy. The results in Table 4.13 were used to visualise two examples in the validation set using Faster R-CNN and VFNet respectively. The examples can be seen in Figure 4.4, where two ground truth examples (4.4a and 4.4b) as well as the results from using the two mentioned models.



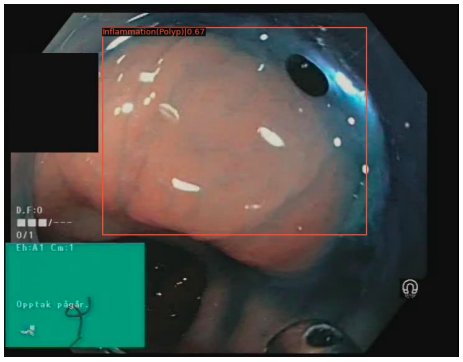
(a) Frame from the new dataset and its corresponding ground truth bounding box.



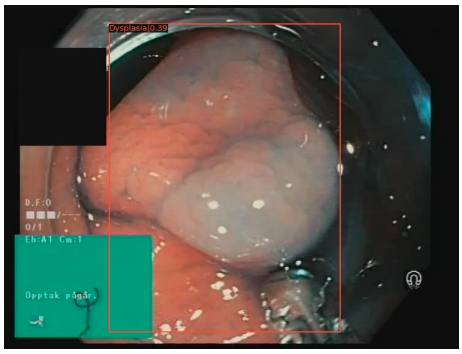
(b) Frame from the new dataset and its corresponding ground truth bounding box.



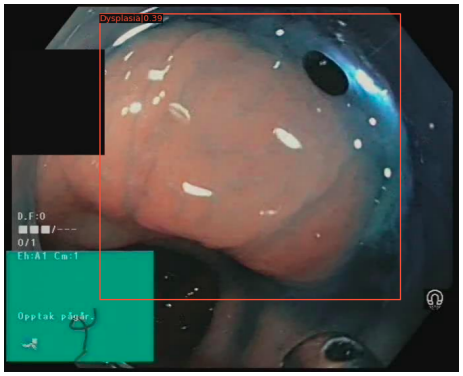
(c) An example of correctly detected hyperplasia using Faster R-CNN, where the model performs quite well in terms of localisation.



(d) An example of detected inflammation (polyp) using Faster R-CNN, where the localisation is done poorly .



(e) An example of incorrectly detected dysplasia using VFNet, where both label and localisation is wrong.



(f) An example of incorrectly detected dysplasia using VFNet, where both label and localisation is wrong.

Figure 4.4: Examples showing the performance of Faster R-CNN and VFNet operating on the new dataset.

As can be seen in Figure 4.4, the area of inflammation bounded in 4.4b is visualised with the human eye and cannot be detected correctly neither by Faster R-CNN nor VFNet. However, Faster R-CNN achieves to detect the area with hyperplasia (Figure 4.4c) even though the bounding box is not as precise compared to the ground truth. The same area is not detected by the VFNet (Figure 4.4e).

RetinaNET was also used, and the results from those measurements are visualised in Table 4.14.

Table 4.14: The mAP scores for the different classes using the model RetinaNET are presented. The AP-values for 0.5 and 0.75 are also visualised.

Class	AP_{50}	AP_{75}	mAP@[0.5 : 0.95]
Dysplasia	0.1	0	0
Hyperplasia	44.4	3.1	13.1
Inflammation (Polyp)	18.9	10.0	9.7
Inflammation (Biopsy)	9.9	3.8	4.2
All classes			6.8

As in the previous case, RetinaNET cannot detect dysplasia, but the rest of the classes can be detected even though the accuracy is quite low in all of the cases. Both Faster R-CNN and RetinaNET achieve the same accuracy of hyperplasia. The main difference is that RetinaNET is slightly better in detecting the inflammation classes. The results of the last experiment are visualised in Table 4.15 using VFNet.

Table 4.15: The mAP values for the different classes using the model VFNet are presented. The AP-values for 0.5 and 0.75 are also visualised.

Class	AP_{50}	AP_{75}	mAP@[0.5 : 0.95]
Dysplasia	0.2	0	0
Hyperplasia	2.7	0.6	1.1
Inflammation (Polyp)	15.6	7.9	9.4
Inflammation (Biopsy)	32.4	1.1	10.4
All classes			5.2

VFNet cannot detect neither dysplasia nor hyperplasia. However, the model is better at detecting the inflammation classes than Faster R-CNN. By comparing VFNet with RetinaNET, it can be seen that VFNet is slightly better at detecting inflammation (biopsy). To sum up, the three different models have quite low accuracy in detecting the four various classes.

4.3.1 Results of Combining the Most Promising Data Augmentation Techniques Using the New Dataset

To evaluate if some models can achieve better accuracy of the new dataset in form of higher mAP scores, pre-trained weights from Kvasir were used. The augmentation techniques in form of rotation, Gaussian blurring and image sharpening filter were used in order to get more data to train the models with. Furthermore, Faster R-CNN was excluded in this stage since it was the model with lowest accuracy in the previous stage.

The training set was augmented by 25% with each technique, and the results using RetinaNET can be seen in Table 4.16.

Table 4.16: The mAP values are presented, where the model RetinaNET and 25% of the augmentation techniques were added. The AP-values for 0.5 and 0.75 are also visualised.

Class	AP_{50}	AP_{75}	mAP@[0.5 : 0.95]
Dysplasia	4.2	0.1	1.1
Hyperplasia	7.0	1.2	2.9
Inflammation (Polyp)	9.3	0.2	2.7
Inflammation (Biopsy)	11.1	9.6	7.4
All classes			3.5

RetinaNet is still quite low in accuracy in form of mAP. Note that all the classes are non-zero in this case, which was not the case in the previous experiments. The same augmentation techniques and amounts as before were used in combination with VFNet, and the results are visualised in Table 4.17.

Table 4.17: The mAP values are presented, where the model VFNet and 25% of the augmentation techniques were added. The AP-values for 0.5 and 0.75 are also visualised.

Class	AP_{50}	AP_{75}	mAP@[0.5 : 0.95]
Dysplasia	0	0	0
Hyperplasia	1.4	0	0.5
Inflammation (Polyp)	10.4	1.0	4.8
Inflammation (Biopsy)	0.2	0	0
All classes			1.3

When augmenting the new dataset with 25%, it can be seen in Table 4.17 that the model achieves lower mAP-scores than the previous experiments in Table 4.15.

The ratio of the different augmentation techniques were increased to 50% in each step. The results from using RetinaNET and pre-trained weigths from Kvasir are visualised in Table 4.18.

Table 4.18: The mAP scores are presented, where the model RetinaNET and 50% of the augmentation techniques were added. The AP-values for 0.5 and 0.75 are also visualised.

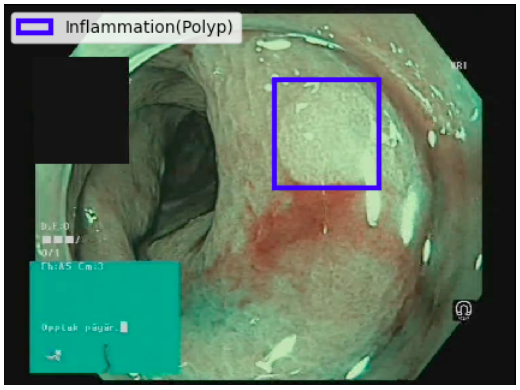
Class	AP_{50}	AP_{75}	mAP@[0.5 : 0.95]
Dysplasia	0.5	0.1	0.2
Hyperplasia	62.5	39.5	37.0
Inflammation (Polyp)	0.5	0	0.1
Inflammation (Biopsy)	0	0	0
All classes			9.3

Observe in Table 4.18 that the mAP for hyperplasia is much higher compared with earlier results. However, the rest of the mAP values are quite low. The results using VFNet can be seen in Table 4.19.

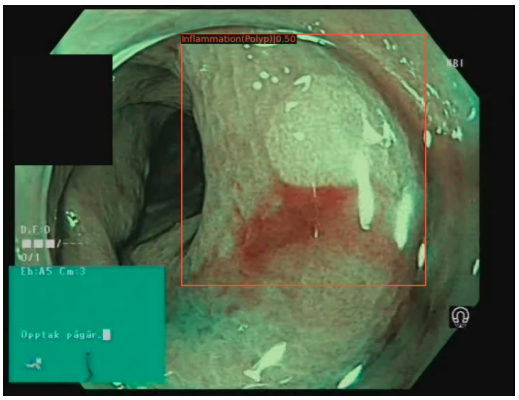
Table 4.19: The mAP values are presented, where the model VFNet and 50% of the augmentation techniques were added. The AP-values for 0.5 and 0.75 are also visualised.

Class	AP_{50}	AP_{75}	mAP@[0.5 : 0.95]
Dysplasia	0.2	0	0.1
Hyperplasia	44.8	27.3	27.9
Inflammation (Polyp)	12.5	3.8	5.8
Inflammation (Biopsy)	3.0	0.8	1.1
All classes			8.7

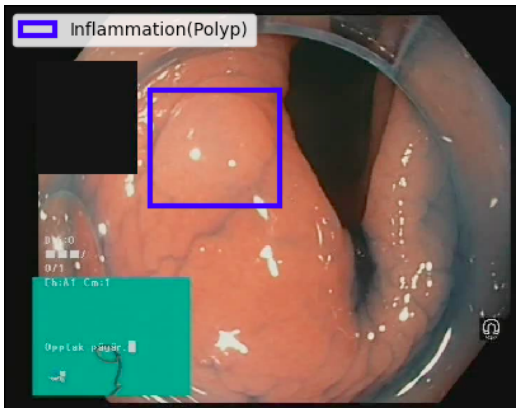
The mAP scores were increased by comparison with the values in Table 4.17. Even in this experiment, the mAP value for the class hyperplasia increased. In order to get some visual results, two examples of images in the validation set were used. The results from Table 4.19 using pre-trained weights from Kvasir can be seen in Figure 4.5, more specifically Figure 4.5b and 4.5d respectively.



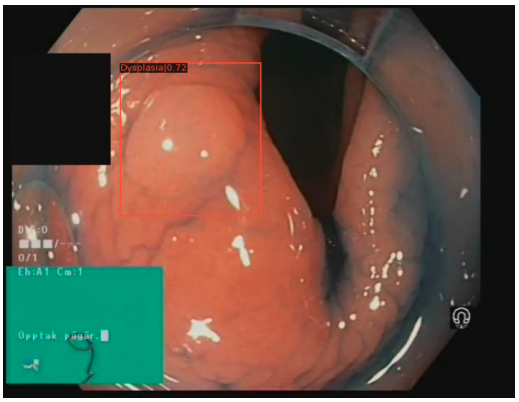
(a) Image from the dataset and its corresponding ground truth bounding box.



(b) Detected inflammation (Polyp), using VFNet. However, the bounding box is too large in comparison with the ground truth.



(c) Image from the dataset and its corresponding ground truth bounding box.



(d) Falsely detected dysplasia in comparison with the ground truth, using VFNet. As visualised, the location is correct but the label is incorrect.

Figure 4.5: Examples showing the performance of VFNet operating on the new dataset.

4.3.2 Results of Combining the Most Promising Data Augmentation Techniques where Biopsy-frames are Removed from the New Dataset

In order to see if the images with the Biopsy-class had negative or positive effect on the learning process, those frames were removed from the dataset. The actual numbers for each class can be seen in Table 4.2. This dataset was used in combination with pre-trained weights from Kvasir and all the augmentation techniques to get more data. The results from using RetinaNET and augmenting 25% of the training set with each technique can be seen in Table 4.20.

Table 4.20: The mAP values are presented, where the model RetinaNET and 25% of the augmentation techniques were added. The AP-values for 0.5 and 0.75 are also visualised.

Class	AP_{50}	AP_{75}	mAP@[0.5 : 0.95]
Dysplasia	0.9	0	0.3
Hyperplasia	9.0	0.1	2.2
Inflammation	32.1	6.1	13.8
All classes			5.4

Note that the mAP for the class inflammation is increased, while the other mAP values are still quite low. VFNet was also used, and the results are visualised in Table 4.21.

Table 4.21: The mAP scores are presented, where the model VFNet and 25% of the augmentation techniques were added. The AP-values for 0.5 and 0.75 are also visualised.

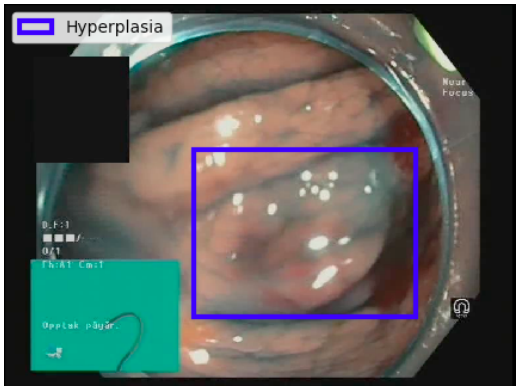
Class	AP_{50}	AP_{75}	mAP@[0.5 : 0.95]
Dysplasia	17.0	1.1	5.5
Hyperplasia	3.6	2.2	1.9
Inflammation	34.4	9.6	14.0
All classes			7.1

Observe that dysplasia achieves higher accuracy in this stage compared to the other experiments. The mAP value for inflammation is also higher than previously, while hyperplasia is quite low. The next step was to augment 50% of the training set with each technique. The results using RetinaNET can be seen in Table 4.22.

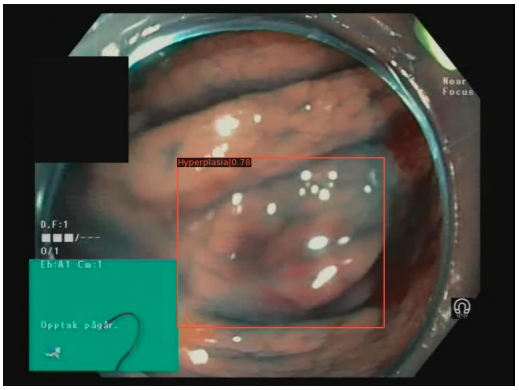
Table 4.22: The mAP values are presented, where the model RetinaNET and 50% of the augmentation techniques were added. The AP-values for 0.5 and 0.75 are also visualised.

Class	AP_{50}	AP_{75}	mAP@[0.5 : 0.95]
Dysplasia	13.9	0.5	4.2
Hyperplasia	29.6	7.4	13.5
Inflammation	4.8	1.9	2.2
All classes			6.7

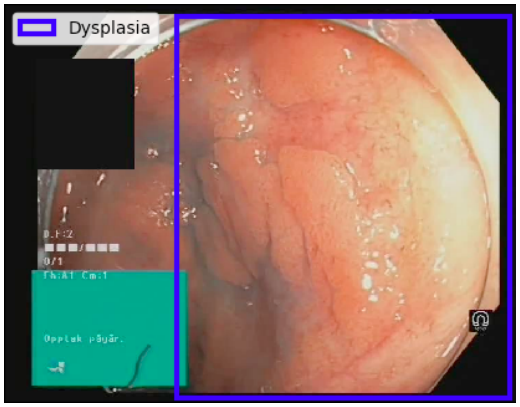
In this experiment, the accuracy of the class hyperplasia is increased in combination with improved accuracy for the class dysplasia. The results from Table 4.22 were used to detect some of the classes. This can be seen by an example in Figure 4.6, where RetinaNET achieves to detect hyperplasia in Figure 4.6b. However, RetinaNET has problem to detect the class dysplasia which can be seen in Figure 4.6d.



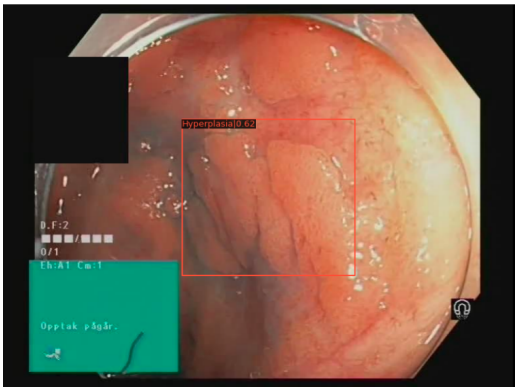
(a) Image from the dataset and its corresponding ground truth bounding box.



(b) Corrected detected hyperplasia using RetinaNET.



(c) Image from the dataset and its corresponding ground truth bounding box.



(d) Falsely detected hyperplasia using RetinaNET. In comparison with its ground truth, both label and bounding box location are detected incorrectly.

Figure 4.6: Examples showing the performance of RetinaNET operating on the new dataset.

VFNet was used as well on the augmented training set. The results can be seen in Table 4.23.

Table 4.23: The mAP values are presented, where the model VFNet and 50% of the augmentation techniques were added. The AP-values for 0.5 and 0.75 are also visualised.

Class	AP_{50}	AP_{75}	mAP@[0.5 : 0.95]
Dysplasia	44.2	0.1	10.7
Hyperplasia	3.0	0	0.6
Inflammation	0.3	0	0.1
All classes			3.8

In this experiment, the highest mAP value for dysplasia is achieved. However, the rest of the classes have still quite low accuracy.

4.4 Comparison with a medical expert

As described in the methodology, a small comparing test was performed in order to evaluate whether there was visual macroscopic difference between the frames in the dataset. The medical student labeled the 50 selected images with a result of 17 correct answers, this corresponds to an accuracy of 34%.

5
Discussion

In this section, the method of the project as well as the results produced is discussed. Lastly the potential of this subject and further work is described.

5.1 Discussion Regarding the Method

The method was planned for receiving the dataset during the project, which lead to a prestudy using Kvasir. The prestudy gave a good base knowledge about computer vision for colonoscopy images, enabling a fast transition to the new dataset when it was received. Since the dataset was finalised later than planned the majority of the time was used for the prestudy on Kvasir. One thing that could have been changed, when looking back on the project, was to test less models and focused more on augmentation techniques even in combination with Kvasir. However, that would have required the knowledge of the small size of the new dataset earlier.

If the dataset had been finalised earlier, it would have been preferred to do less tests of models on Kvasir and focused more on improving the dataset. The dataset in itself could have been improved by sorting the different images to eliminate equal images from nearby frames. Further on, the dataset could have been artificially improved by more complex augmentation methods like mixing images, adding random noise-patches or using Generative Adversarial Network (GAN) that generates new artificial images.

5.2 Discussion Regarding the Results

The majority of the models performed quite well on Kvasir, particularly Faster R-CNN, RetinaNET and VFNet which achieved a mAP score higher than 70%. However, Kvasir contains 1000 images of unique frames with only one class, which makes it a well designed dataset with great variation within the findings. The results indicates that some of the object detection models have great potential in detecting objects from such surrounding, more specifically in the colon. The three investigated augmentation techniques gave slight improvements for the different models in different cases. For instance, in Table 4.6, 4.8 and 4.11, there are either the same values of the mAP or slightly better compared to the results presented in Table 4.3. There are no drastic differences between those values, which leads to an uncertainty whether the augmentation techniques lead to a significant difference or if the models already have saturated on this dataset.

It can also be noted that the mAP values in Table 4.9 and 4.12, presenting the results of 75% blurring and sharpening, are lower than the values of Table 4.8 and 4.11, presenting 50% augmentation of the same techniques. This is an unexpected result that could have been caused by the saturation of the models. This trend is not seen when using rotation (Table 4.5 and 4.6), which could indicate that rotations gives a more diverse augmented dataset in this case. However, it can still be evaluated that VFNet achieved a maximum accuracy in form of mAP of about 80%. Even though the numbers are not directly comparable, the miss-rate done by physicians in real life is around 23%. This could mean that VFNet achieved nearly the same accuracy, which may indicate future use for similar contexts.

Moreover, the newly created dataset was investigated with Faster R-CNN, RetinaNET and VFNet at first, which then was further investigated with only RetinaNET and VFNet. To summarise the different experiments, it can be evaluated that the models are quite unreliable in predicting the different classes. There are various aspects which may have been the cause of these results. First of all, the dataset is not made of unique findings which is the case in Kvasir. This leads to similar frames in most of the cases, which can result in correlated data. From Figure 4.4b, the area of inflammation is visible even though the mAP values are quite low. The explanation might be that the data is correlated which can lead to overfitted models. Otherwise, object detection models might handle similar types of problems.

Secondly, the actual numbers of images in the dataset are quite small and extracted from only ten colonoscopy videos, which also lead to small amount of data for each class respectively. This can be compared to Kvasir, where 1000 images representing only one class. Additionally, the quality of some of the frames is quite poor, making it more difficult for the models to detect visual differences between each frame and finding.

Lastly, in the creation of the dataset, biopsies were marked at a first stage. Then, the biopsies were paired and checked with the microscopic investigations in order to give the correct label. Since the network is doing macroscopic investigations, there must be a visual difference in the macroscopic image. Therefore, it is easier for the models to detect those changes if there are structural differences within the frames. Instead, the majority of the frames are decided from those microscopic investigations, which means that differences between the bounding boxes and the surroundings can be very difficult to detect even with human eyes. Even though it is believed that the lack of structural changes are the cause the bad performances, Figure 4.4 indicates that there are problems detecting visual changes. As mentioned earlier, in Figure 4.4b, there are an area of inflammation visible for the human eyes. However, the models lack in performance to detect those areas which indicates that there are weak aspects within the creation of the dataset. This weak aspects can be uncertain areas and the few unique findings.

The aspects in the previous paragraph may have resulted in the low precision values

that are presented in Table 4.13, 4.14 and 4.15. Therefore, evaluating the models' performances on this dataset is a complex task. The same situation remains when the augmentation techniques are added to the training set. In some cases when the dataset was augmented with 50% of each technique, *e.g.* in Table 4.18 and 4.19, a slight increase in mAP is visualised since hyperplasia can be detected with a higher accuracy. Albeit, the increase in detecting hyperplasia was still quite low and could therefore be misleading. Additionally, the rest of the classes' accuracy are decreased compared with Table 4.14 and 4.15 respectively.

In the last case when the Biopsy-frames were removed from the dataset, it can be seen that dysplasia is detected with non-zero accuracy in three cases out of four comparing with the results on the new dataset. Using RetinaNET in combination with pre-trained weights from Kvasir and 50% augmented data with each technique (Table 4.22) shows higher accuracy in detecting dysplasia (compared with Table 4.14). However, hyperplasia is detected with lower accuracy in the new case compared with Table 4.14. This is not optimal in medical purposes, since it is essential for such detecting systems to distinguish between dysplasia and hyperplasia with high sensitivity and specificity. The effect of not being able to differentiate between those two classes can be devastating for the patient.

When increasing the percentage augmented on the new dataset without the biopsy frames, it could be seen that the total mAP is almost increased for all cases besides for VFNet in Table 4.23. This is not an expected behaviour, and is thought to be due to the small size and variation of the dataset. All images are similar and when augmentation is used to increase the number of images there is a risk of getting more similar images when the selection is done randomly. In a worst case scenario, all images augmented could have come from only a few lesions which would not add much more diversity to the dataset. Therefore, the dataset needs to be re-designed with unique finding in each frame before the purpose of the study can be rejected or not.

The benchmarking test showed that it is quite difficult to see microscopic differences on this dataset by human eyes. This means that it is very challenging for the neural networks to see the same things. The test showed that a medically trained eye cannot distinguish between the different classes resulting in the same accuracy as the coincidence.

5.3 Future Work

To further investigate the potential of computer vision as an aid in colonoscopy investigations, a better dataset will be needed. When comparing the new dataset with Kvasir the main difference is that Kvasir contains one image per unique finding, collected over several years from many patients, while the new dataset has many similar images from neighbouring frames. The duplication of images needs to be sorted out and the frames needs to be more carefully collected and selected. When looking at the collection of the data more improvements can be done. Images with

higher resolution will give better data for the detection, as well as stopping and focusing on the positions of interest before taking the biopsy to give clearer and more focused images. With a better dataset further investigations can be done. Besides testing object detection models like in this study, classification models could be tested. This could evaluate if it would be easier to only identify the label of the image and exclude the localization task. Even with a better dataset, artificial improvements on the dataset can also be done to further improve the results, as mentioned before GAN, noise patches and mixing could have been tested.

6
Conclusion

This study, in the results from using Kvasir has shown potential of the usage of object detection in colonoscopic images. With the material in the new dataset and its quality, a conclusion about the potential cannot be drawn. A new dataset is needed to test how well object detection could perform on this type of data, since the positive results from Kvasir indicate that it is worth testing. The largest query is the difficulty for the model to see the microscopic changes in the images on macroscopic level, which would need a high resolution and high quality dataset to investigate if possible.

Bibliography

[1] Konstantin Pogorelov et al. "KVASIR: A Multi-Class Image Dataset for Computer Aided Gastrointestinal Disease Detection". In: *Proceedings of the 8th ACM on Multimedia Systems Conference*. MMSys'17. [Online]. New York, NY, USA: ACM, 2017, pp. 164–169. ISBN: 978-1-4503-5002-0. DOI: 10.1145/3083187.3083212. URL: http://doi.acm.org/10.1145/3083187.3083212 (visited on 17/03/2021).

[2] Junfeng Gao et al. "Computer Vision in Healthcare Applications". In: *J Healthc Eng* (2018). [Online]. DOI: 10.1155/2018/5157020. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5857319/ (visited on 30/03/2021).

[3] Dominic King et al. "Changing patterns in the epidemiology and outcomes of inflammatory bowel disease in the United Kingdom: 2000-2018." In: *Aliment Pharmacol Ther* 51.10 (2020). [Online], pp. 922–4. DOI: https://doi.org/10.1111/apt.15701. (Visited on 25/01/2021).

[4] Jack N. Cadwaller. *Crohn's Disease: Etiology, Pathogenesis and Interventions*. [Online]. Nova Science Publishers, 2008, p. vii. ISBN: 9781604569933. URL: https://ebookcentral.proquest.com/lib/chalmers/detail.action?docID=3020429. (visited on 26/02/2021).

[5] Mayo Clinic Staff. *Chron's disease*. [Online]. Oct. 2020. URL: https://www.mayoclinic.org/diseases-conditions/crohns-disease/symptoms-causes/syc-20353304 (visited on 26/01/2021).

[6] S. Mehta and Kar. Ashutosh. *Pharmaceutical Pharmacology - 12.2 Ulcerative Colitis (UC)*. [Online]. New Academic Science, 2014, pp. 953–954. ISBN: 978-1-781830-68-0. URL: https://app.knovel.com/hotlink/khtml/id:kt011NOPQ4/pharmaceutical-pharmacology/ulcerative-colitis-uc (visited on 26/02/2021).

[7] Mayo Clinic Staff. *Ulcerative colitis*. [Online]. Oct. 2020. URL: https://www.mayoclinic.org/diseases-conditions/ulcerative-colitis/symptoms-causes/syc-20353326 (visited on 26/01/2021).

[8] Ola Olén et al. "Colorectal cancer in Crohn's disease: a Scandinavian population-based cohort study". In: *The Lancet* 5.5 (2020). [Online], pp. 475–484. DOI: https://doi.org/10.1016/S2468-1253(20)30005-4. (Visited on 25/01/2021).

[9] Ola Olén et al. “Colorectal cancer in ulcerative colitis: a Scandinavian population-based cohort study”. In: *The Lancet* 395.10218 (2020). [Online], pp. 123–131. ISSN: 0140-6736. DOI: https://doi.org/10.1016/S0140-6736(19)32545-0. URL: http://www.sciencedirect.com/science/article/pii/S0140673619325450 (visited on 25/01/2021).

[10] Freddie Bray et al. “Global cancer statistics 2018: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries”. In: *CA: A Cancer Journal for Clinicians* 68.6 (2018). [Online], pp. 394–424. DOI: https://doi.org/10.3322/caac.21492. eprint: https://acsjournals.onlinelibrary.wiley.com/doi/pdf/10.3322/caac.21492. URL: https://acsjournals.onlinelibrary.wiley.com/doi/abs/10.3322/caac.21492 (visited on 18/02/2021).

[11] A. Pera et al. “Colonoscopy in inflammatory bowel disease: Diagnostic accuracy and proposal of an endoscopic score”. In: *Gastroenterology* 92.1 (1987). [Online], pp. 181–185. ISSN: 0016-5085. DOI: https://doi.org/10.1016/0016-5085(87)90856-0. URL: http://www.sciencedirect.com/science/article/pii/0016508587908560 (visited on 26/01/2021).

[12] Laurent Beaugerie et al. “Risk of Colorectal High-Grade Dysplasia and Cancer in a Prospective Observational Cohort of Patients With Inflammatory Bowel Disease”. In: *Gastroenterology* 145.1 (2013). [Online], 166–175.e8. ISSN: 0016-5085. DOI: https://doi.org/10.1053/j.gastro.2013.03.044. URL: http://www.sciencedirect.com/science/article/pii/S0016508513004587 (visited on 26/01/2021).

[13] Nam Hee Kim et al. “Miss rate of colorectal neoplastic polyps and risk factors for missed polyps in consecutive colonoscopies”. In: *Intest Res* 15.3 (2017). [Online], pp. 411–418. URL: 10.5217/ir.2017.15.3.411 (visited on 26/01/2021).

[14] Arnaud Setio et al. “Pulmonary Nodule Detection in CT Images: False Positive Reduction Using Multi-View Convolutional Networks”. In: *IEEE Transactions on Medical Imaging* 35 (Mar. 2016). [Online], pp. 1–1. DOI: 10.1109/TMI.2016.2536809. (Visited on 25/01/2021).

[15] Alessandro Repici et al. “Efficacy of Real-Time Computer-Aided Detection of Colorectal Neoplasia in a Randomized Trial”. In: *Gastroenterology* 159.2 (2020). [Online], 512–520.e7. ISSN: 0016-5085. DOI: https://doi.org/10.1053/j.gastro.2020.04.062. URL: http://www.sciencedirect.com/science/article/pii/S0016508520305837 (visited on 25/01/2021).

[16] Dayong Wang et al. *Deep Learning for Identifying Metastatic Breast Cancer*. [Online]. 2016. arXiv: 1606.05718 [q-bio.QM]. (Visited on 25/01/2021).

[17] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. Available: http://www.deeplearningbook.org. [Online]. MIT Press, 2016. (Visited on 25/01/2021).

[18] Yali Amit, Pedro Felzenszwalb and Ross Girshick. “Object Detection”. In: *Computer Vision: A Reference Guide*. Cham: Springer International Publishing, 2020, pp. 1–9. ISBN: 978-3-030-03243-2. DOI: 10.1007/978-3-030-03243-2_660-1. URL: https://doi.org/10.1007/978-3-030-03243-2_660-1.

[19] Jamie Shotton and Pushmeet Kohli. "Semantic Image Segmentation: Traditional Approach". In: *Computer Vision: A Reference Guide*. Cham: Springer International Publishing, 2019, pp. 1–4. ISBN: 978-3-030-03243-2. DOI: 10.1007/978-3-030-03243-2_251-1. URL: https://doi.org/10.1007/978-3-030-03243-2_251-1.

[20] Anna Bendt. *Ulcerös kolit*. [Online]. Aug. 2020. URL: https://www.1177.se/Vastra-Gotaland/sjukdomar--besvar/mage-och-tarm/tarmsjukdomar/ulceros-kolit/.

[21] Susanna Schultz. *Vad är cancer?* [Online]. June 2020. URL: https://www.1177.se/Vastra-Gotaland/sjukdomar--besvar/cancer/fakta-om-cancer/vad-ar-cancer/.

[22] Susanna Schultz. *Vad är cancer?* [Electronic image]. June 2020. URL: https://www.1177.se/Vastra-Gotaland/sjukdomar--besvar/cancer/fakta-om-cancer/vad-ar-cancer/.

[23] Mark T Keane and Eoin M Kenny. *How Case Based Reasoning Explained Neural Networks: An XAI Survey of Post-Hoc Explanation-by-Example in ANN-CBR Twins*. [Online]. 2019. arXiv: 1905.07186 [cs.AI]. URL: https://arxiv.org/abs/1905.07186? (visited on 02/03/2021).

[24] Gerard J. Tortora and Bryan Derrickson. *Introduction to the human body*. 10th ed. Wiley, 2015, pp. 228, 236. ISBN: 978-1-118-58318-0.

[25] A. K. Jain, Jianchang Mao and K. M. Mohiuddin. "Artificial neural networks: a tutorial". In: *Computer* 29.3 (1996). [Online], pp. 31–44. DOI: 10.1109/2.485891. (Visited on 03/03/2021).

[26] Renanar2. *ReLU and Nonnegative Soft Thresholding Functions.svg*. [Electronic image]. URL: https://search.creativecommons.org/photos/8e35394e-5b00-4560-8160-1b58909e6452 (visited on 03/03/2021).

[27] Hamed Habibi Aghdam and Elnaz Jahani Heravi. "Convolutional Neural Networks". In: *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification*. [Online]. Cham: Springer International Publishing, 2017, pp. 85–130. ISBN: 978-3-319-57550-6. DOI: 10.1007/978-3-319-57550-6_3. URL: https://doi.org/10.1007/978-3-319-57550-6_3 (visited on 18/03/2021).

[28] Interior Design Best. *Convolutional Neural Network Architecture*. [Electronic image]. 2018. URL: https://www.flickr.com/photos/166111585@N05/43080092915/in/photolist-28CQJJF-2iQ5rgh-2j7STwV-pFoqus-Vu6tAT-DWdDiD-zTK9Us-Ag3aXL-WE1C1J-2iFPEKQ-zmrvD8-WHA6kZ-zUDHVy-21N3Pub-WvXgvp-Vu6tWH-WHA6PV-WHA91P-2hUumza-WE1GgN-WvXsZX-WvXgpn-Vro629-WHA7hi-WvXfUK-WvXmxt-MqF1iE-WsBisQ-WsBfTG-Vro6Ms-Vu6ymz-WvXjEv-WvXgfe-WHA7We-WE1G1h-CzVWZ1-23Gc14W-23Gc1Su-23Gc1K5-26rUQn4-2hswY6B-2hsvZ6i-TEGq2s-2hswXVr-26rUNpB-23Gc1NS-26rUPtR-LESS4c-WvXkbR-WE1Hcf (visited on 18/03/2021).

[29] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *Computer Vision – ECCV 2014*. [Online]. Cham: Springer International Publishing, 2014, pp. 818–833. ISBN: 978-3-319-10590-1. (Visited on 18/03/2021).

[30] Matthew Stewart. *The convolution operation*. [Electronic image]. 2019. URL: https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac (visited on 18/03/2021).

[31] Keiron O'Shea and Ryan Nash. *An Introduction to Convolutional Neural Networks*. [Online]. 2015. arXiv: 1511.08458 [cs.NE]. URL: https://arxiv.org/pdf/1511.08458.pdf (visited on 19/03/2021).

[32] Kartikeya Rana. *Max Pooling layer applied a single slice of an imput volume*. [Electronic image]. 2020. URL: https://ai.plainenglish.io/pooling-layer-beginner-to-intermediate-fa0dbdce80eb (visited on 19/03/2021).

[33] J. Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. DOI: 10.1109/CVPR.2016.91.

[34] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. [Online]. 2018. (Visited on 20/03/2021).

[35] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. [Electronic image]. 2018. (Visited on 20/03/2021).

[36] Alexey Bochkovskiy, Chien-Yao Wang and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. [Online]. 2020. (Visited on 13/03/2021).

[37] Chien-Yao Wang et al. *CSPNet: A New Backbone that can Enhance Learning Capability of CNN*. [Online]. 2019. (Visited on 14/03/2021).

[38] Kaiming He et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *Lecture Notes in Computer Science* (2014). [Online], pp. 346–361. ISSN: 1611-3349. DOI: 10.1007/978-3-319-10578-9_23. URL: http://dx.doi.org/10.1007/978-3-319-10578-9_23 (visited on 23/03/2021).

[39] Kaiming He et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *Lecture Notes in Computer Science* (2014). [Electronic image], pp. 346–361. ISSN: 1611-3349. DOI: 10.1007/978-3-319-10578-9_23. URL: http://dx.doi.org/10.1007/978-3-319-10578-9_23.

[40] Shu Liu et al. *Path Aggregation Network for Instance Segmentation*. [Online]. 2018. (Visited on 24/03/2021).

[41] Shaoqing Ren et al. *Region Proposal Network (RPN)*. [Electronic image]. 2016. URL: https://arxiv.org/pdf/1506.01497.pdf (visited on 23/03/2021).

[42] Tsung-Yi Lin et al. *Feature Pyramid Networks for Object Detection*. [Online]. 2017. arXiv: 1612.03144 [cs.CV]. URL: https://arxiv.org/pdf/1612.03144.pdf (visited on 25/03/2021).

[43] Tsung-Yi Lin et al. *Feature Pyramid Networks for Object Detection*. [Electronic image]. 2017. (Visited on 25/03/2021).

[44] Mingxing Tan, Ruoming Pang and Quoc V. Le. *EfficientDet: Scalable and Efficient Object Detection*. [Online]. 2020. (Visited on 25/03/2021).

[45] Ross Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. [Online]. 2014. arXiv: 1311.2524 [cs.CV]. URL: https://arxiv.org/pdf/1311.2524.pdf (visited on 22/03/2021).

[46] Ross Girshick et al. *R-CNN: Regions with CNN features*. [Electronic image]. 2014. URL: https://arxiv.org/pdf/1311.2524.pdf (visited on 22/03/2021).

[47] Pedro F Felzenszwalb and Daniel P Huttenlocher. "Efficient Graph-Based Image Segmentation". In: *International Journal of Computer Vision* 59 (2004). [Online], pp. 167–181. DOI: https://doi.org/10.1023/B:VISI.0000022288.19776.77. (Visited on 22/03/2021).

[48] J. R. R Uijlings et al. "Selective Search for Object Recognition". In: *International Journal of Computer Vision* 104 (2013). [Online], pp. 154–171. DOI: https://doi.org/10.1007/s11263-013-0620-5. (Visited on 22/03/2021).

[49] Ross Girshick. *Fast R-CNN*. [Online]. 2015. arXiv: 1504.08083 [cs.CV]. URL: https://arxiv.org/pdf/1504.08083.pdf (visited on 22/03/2021).

[50] Ross Girshick. *Fast R-CNN architecture*. [Electronic image]. 2015. URL: https://arxiv.org/pdf/1504.08083.pdf (visited on 22/03/2021).

[51] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. [Online]. 2016. arXiv: 1506.01497 [cs.CV]. URL: https://arxiv.org/pdf/1506.01497.pdf (visited on 23/03/2021).

[52] Shaoqing Ren et al. *Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network*. [Electronic image]. 2016. URL: https://arxiv.org/pdf/1506.01497.pdf (visited on 23/03/2021).

[53] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. [Electronic image]. 2020. (Visited on 29/03/2021).

[54] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. [Online]. 2020. (Visited on 29/03/2021).

[55] Mingxing Tan, Ruoming Pang and Quoc V. Le. *EfficientDet: Scalable and Efficient Object Detection*. [Electronic image]. 2020. (Visited on 25/03/2021).

[56] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. [Online]. 2018. arXiv: 1708.02002 [cs.CV]. URL: https://arxiv.org/pdf/1708.02002.pdf (visited on 24/03/2021).

[57] Tsung-Yi Lin et al. *The one-stage RetinaNet network architecture uses a Feature Pyramid Network (FPN) backbone on top of a feedforward ResNet architecture*. [Electronic image]. 2018. URL: https://arxiv.org/pdf/1708.02002.pdf (visited on 24/03/2021).

[58] Nicolas Carion et al. *End-to-End Object Detection with Transformers*. [Online]. 2020. arXiv: 2005.12872 [cs.CV]. URL: https://arxiv.org/pdf/2005.12872.pdf (visited on 24/03/2021).

[59] Nicolas Carion et al. *DETR uses a conventional CNN backbone to learn a 2D representation of an input image*. [Electronic image]. 2020. URL: https://arxiv.org/pdf/2005.12872.pdf (visited on 24/03/2021).

[60] Ashish Vaswani et al. *Attention Is All You Need*. [Online]. 2017. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/pdf/1706.03762.pdf (visited on 29/03/2021).

[61] Haoyang Zhang et al. *VarifocalNet: An IoU-aware Dense Object Detector*. [Online]. 2021. arXiv: 2008.13367 [cs.CV]. URL: https://arxiv.org/pdf/2008.13367v2.pdf (visited on 16/04/2021).

[62] Haoyang Zhang et al. *VarifocalNet: An IoU-aware Dense Object Detector*. [Electronic image]. 2020. URL: https://arxiv.org/pdf/2008.13367v2.pdf (visited on 06/05/2021).

[63] Jifeng Dai et al. *Deformable Convolutional Networks*. 2017. arXiv: 1703.06211 [cs.CV].

[64] Philip May. *Improved Image Augmentation for Convolutional Neural Networks by Copyout and CopyPairing*. [Online]. 2019. arXiv: 1909.00390 [cs.CV]. URL: https://arxiv.org/pdf/1909.00390.pdf (visited on 21/04/2021).

[65] Connor Shorten and Taghi M Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning". In: *Journal of Big Data* 6.60 (2019). [Online]. DOI: 10.1186/s40537-019-0197-0. URL: https://link.springer.com/article/10.1186/s40537-019-0197-0?code=a6ae644c-3bfc-43d9-b292-82d77d5890d5#Sec3 (visited on 26/04/2021).

[66] David C. Lay, Steven R. Lay and Judi J. McDonald. *Linear Algebra and Its Applications*. 5th ed. [Online]. Pearson, 2016, p. 142. URL: https://math.berkeley.edu/~yonah/files/Linear%5C%20Algebra.pdf (visited on 07/05/2021).

[67] Rafael C Gonzalez and Richard E Woods. *Digital Image Processing*. 4th ed. Pearson, 2018, pp. 167, 179. ISBN: 10:1-292-22304-9.

[68] AlexeyAB. *YOLOv4 / Scaled-YOLOv4 / YOLO - Neural Networks for Object Detection*. [Online]. URL: https://github.com/AlexeyAB/darknet (visited on 17/03/2021).

[69] OpenMMLab. *OpenMMLab Detection Toolbox and Benchmark*. [Online]. URL: https://github.com/open-mmlab/mmdetection (visited on 17/03/2021).

[70] Tensorflow. *Tensorflow*. [Online]. URL: https://github.com/tensorflow/tensorflow (visited on 17/03/2021).

[71] cocodataset. *cocodataset.github.io*. [Online]. URL: https://github.com/cocodataset/cocodataset.github.io (visited on 17/03/2021).

[72] Debesh Jha et al. "Kvasir-seg: A segmented polyp dataset". In: *International Conference on Multimedia Modeling*. [Electronic image]. Springer. 2020, pp. 451–462.

A

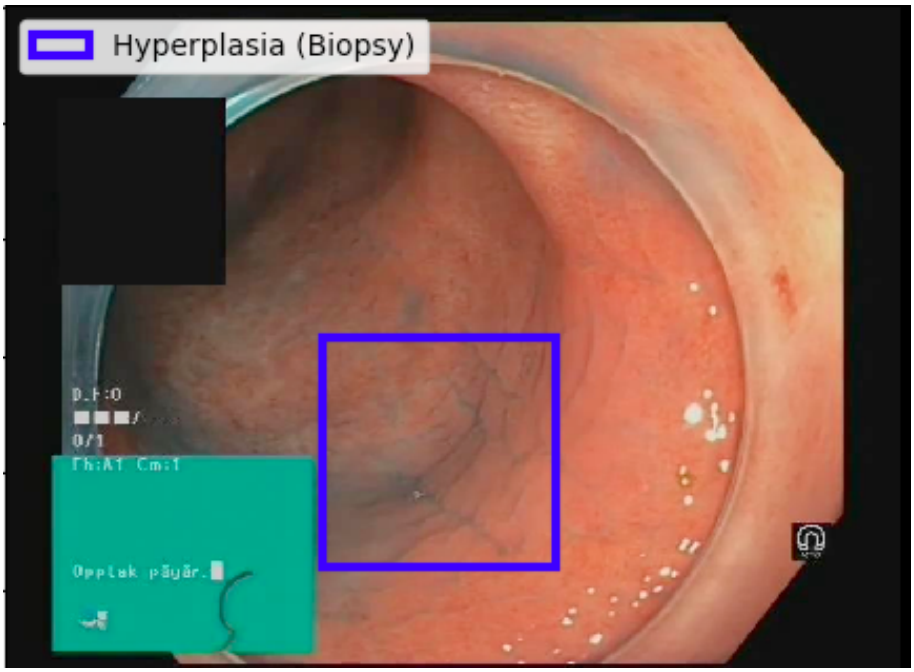
Appendix 1 - Images From The New Dataset

A.1 Images

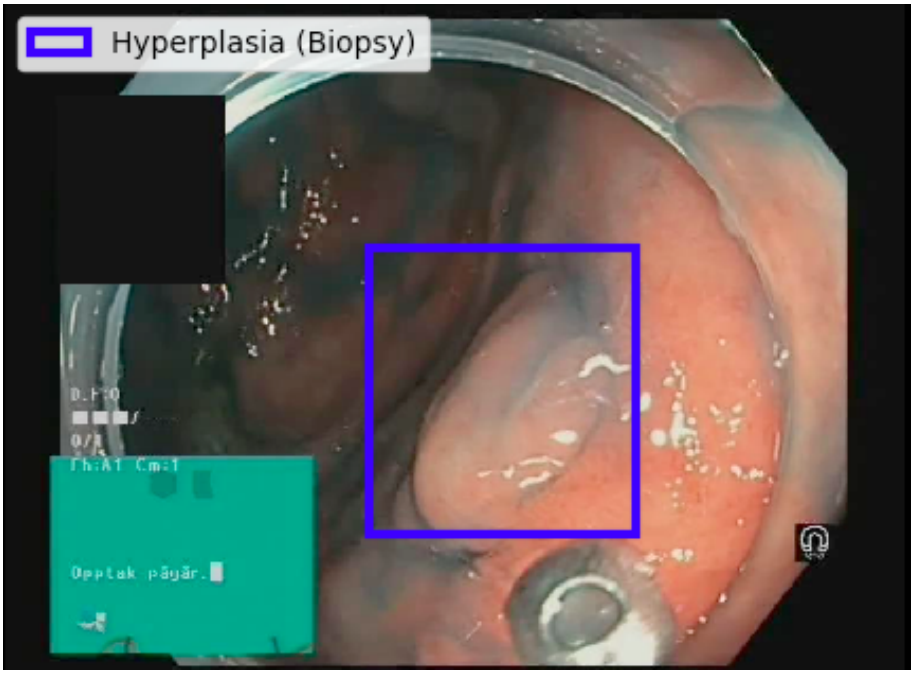
In this section, some examples of images from the newly created dataset will be presented. As described in the introduction, six classes were used at first. More specifically, the classes are listed below.

- Hyperplasia (Biopsy)
- Hyperplasia (Polyp)
- Dysplasia (Biopsy)
- Dysplasia (Polyp)
- Inflammation (Biopsy)
- Inflammation (Polyp)

In Figure A.1, two examples of images with Hyperplasia (Biopsy) are visualised.



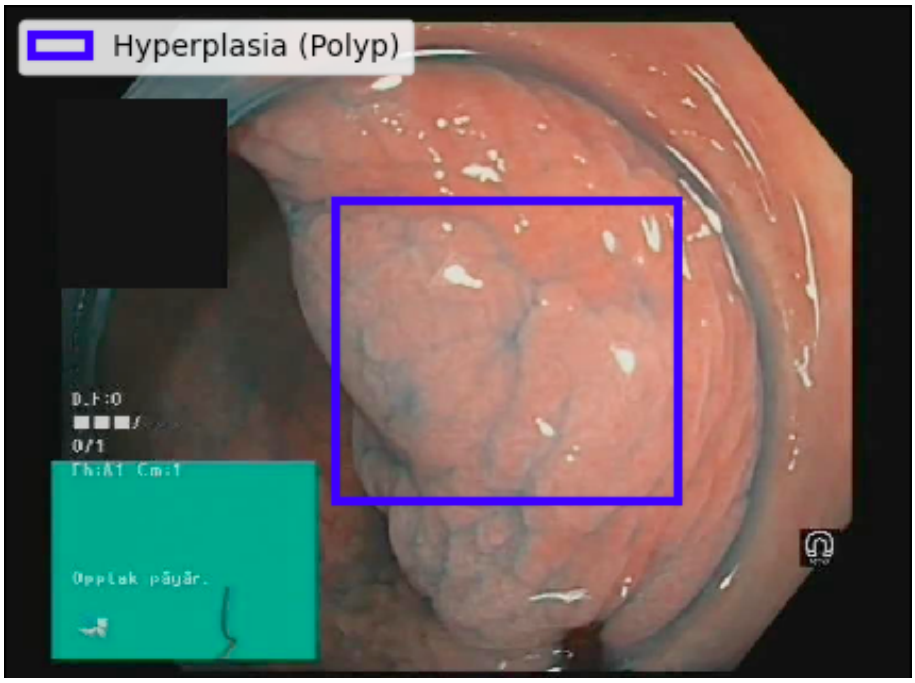
(a) An example of an image in the new dataset visualised with its corresponding bounding box and label (Hyperplasia (Biopsy)).



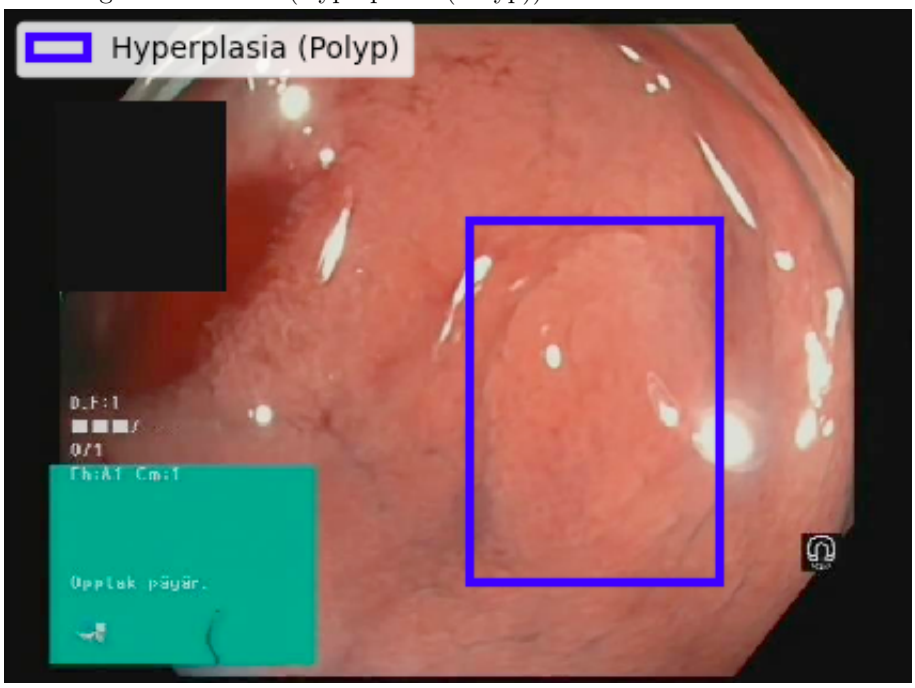
(b) Another example of an image in the new dataset visualised with its corresponding bounding box and label (Hyperplasia (Biopsy)).

Figure A.1: Two examples of images with Hyperplasia (Biopsy).

In Figure A.2, two examples of images with Hyperplasia (Polyp) are presented.



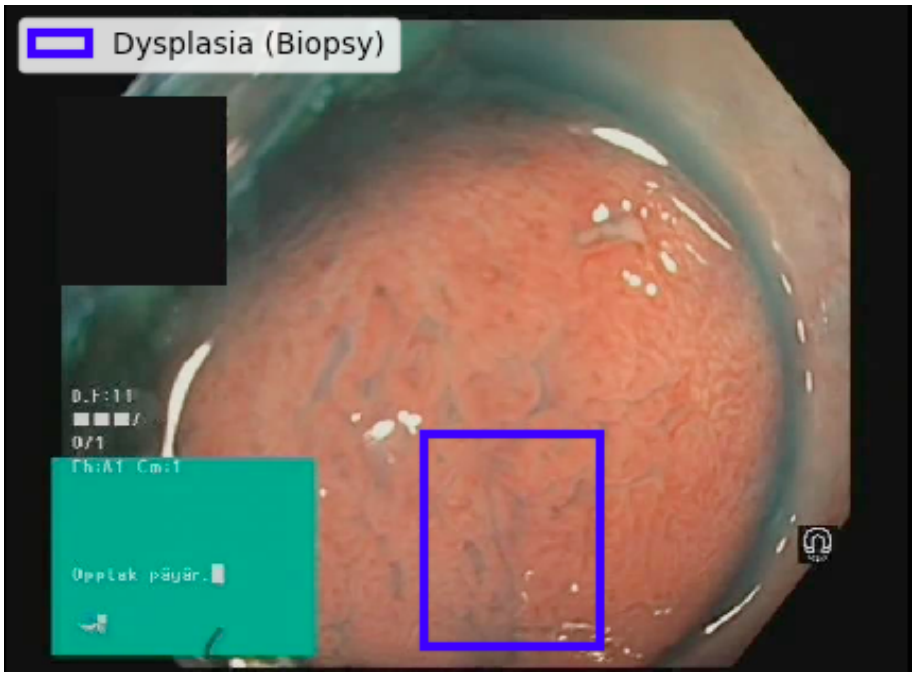
(a) An example of an image in the new dataset visualised with its corresponding bounding box and label (Hyperplasia (Polyp)).



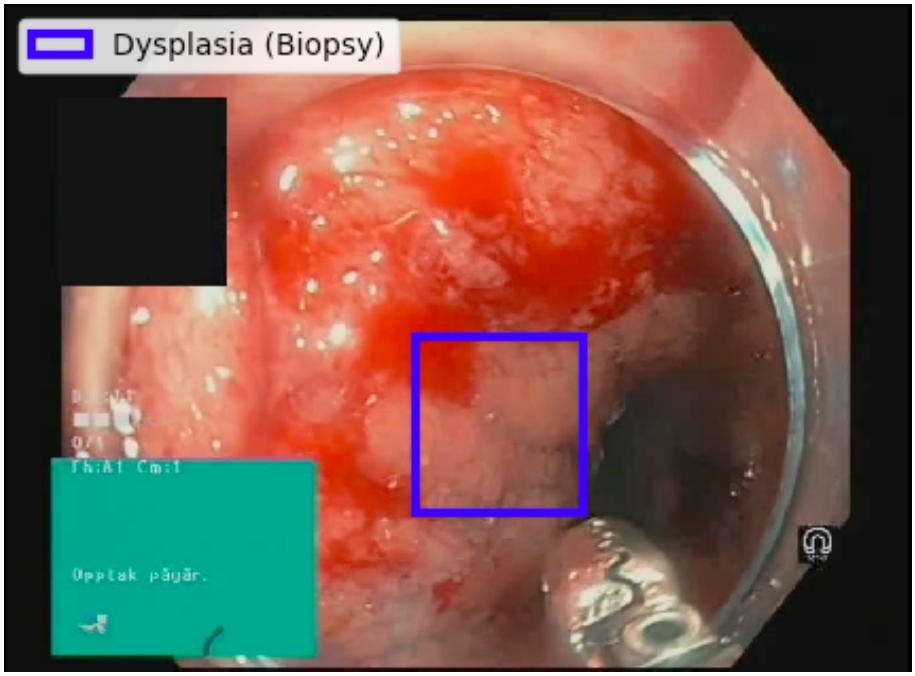
(b) Another example of an image in the new dataset visualised with its corresponding bounding box and label (Hyperplasia (Polyp)).

Figure A.2: Two examples of images with Hyperplasia (Polyp).

In Figure A.3, two examples of images with Dysplasia (Biopsy) are visualised.



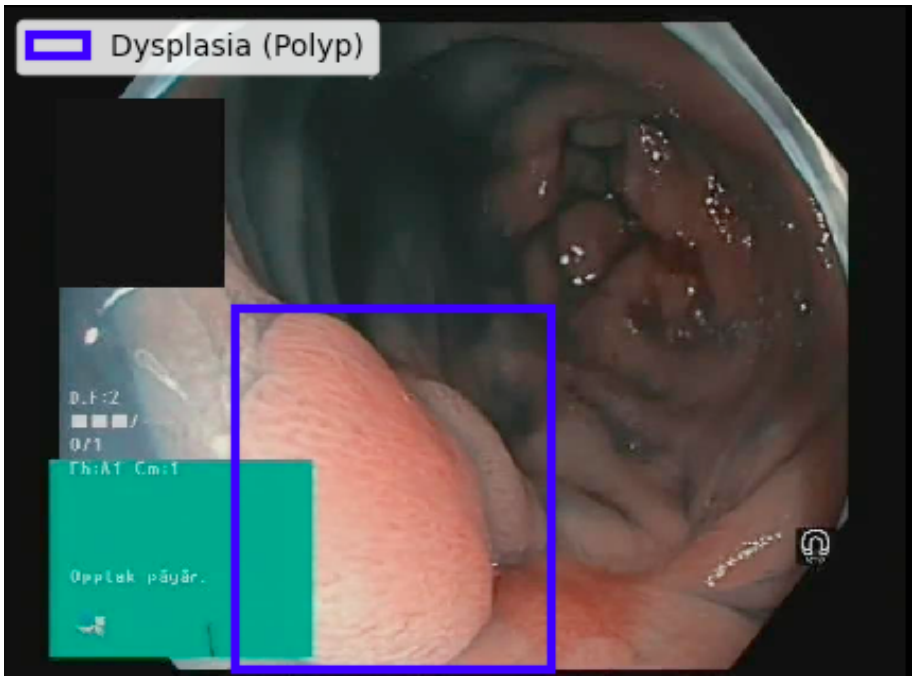
(a) An example of an image in the new dataset visualised with its corresponding bounding box and label (Dysplasia (Biopsy)).



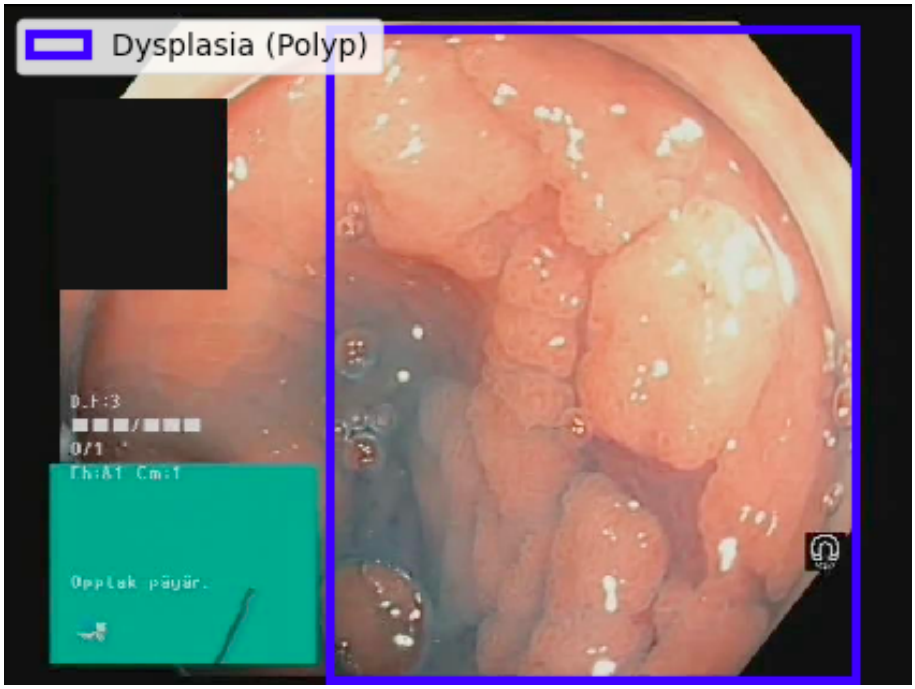
(b) Another example of an image in the new dataset visualised with its corresponding bounding box and label (Dysplasia (Biopsy)).

Figure A.3: Two examples of images with Dysplasia (Biopsy).

In Figure A.4, two examples of images with Dysplasia (Polyp) can be seen.



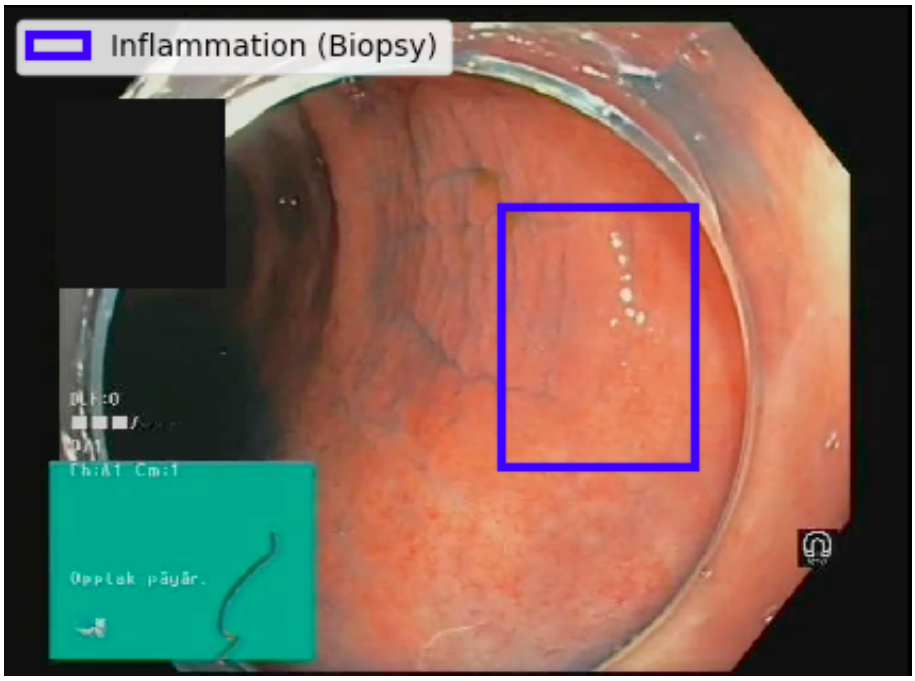
(a) An example of an image in the new dataset visualised with its corresponding bounding box and label (Dysplasia (Polyp)).



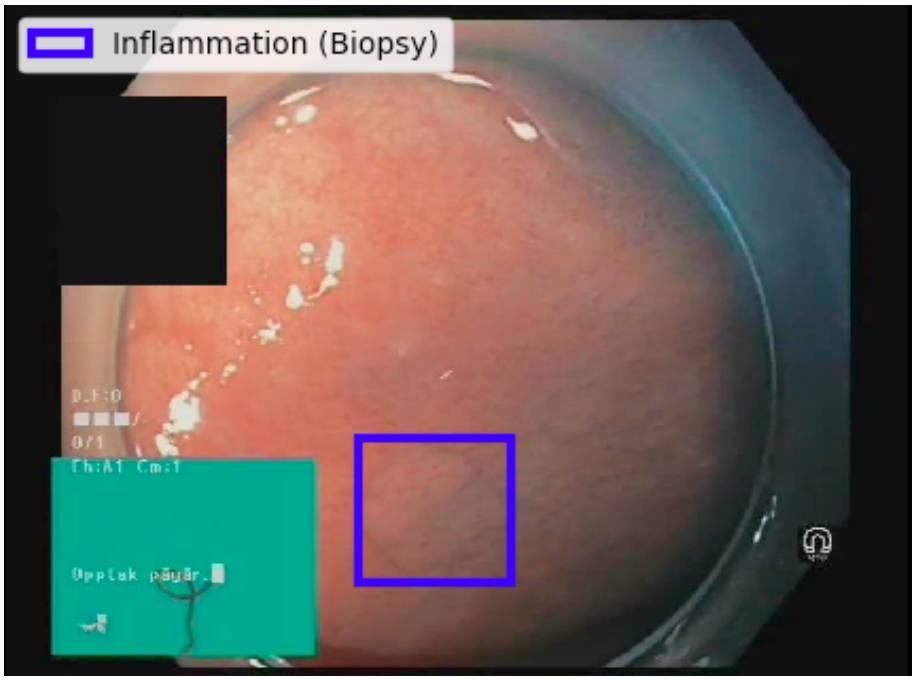
(b) Another example of an image in the new dataset visualised with its corresponding bounding box and label (Dysplasia (Polyp)).

Figure A.4: Two examples of images with Dysplasia (Polyp).

In Figure A.5, two examples of images with Inflammation (Biopsy) are presented.



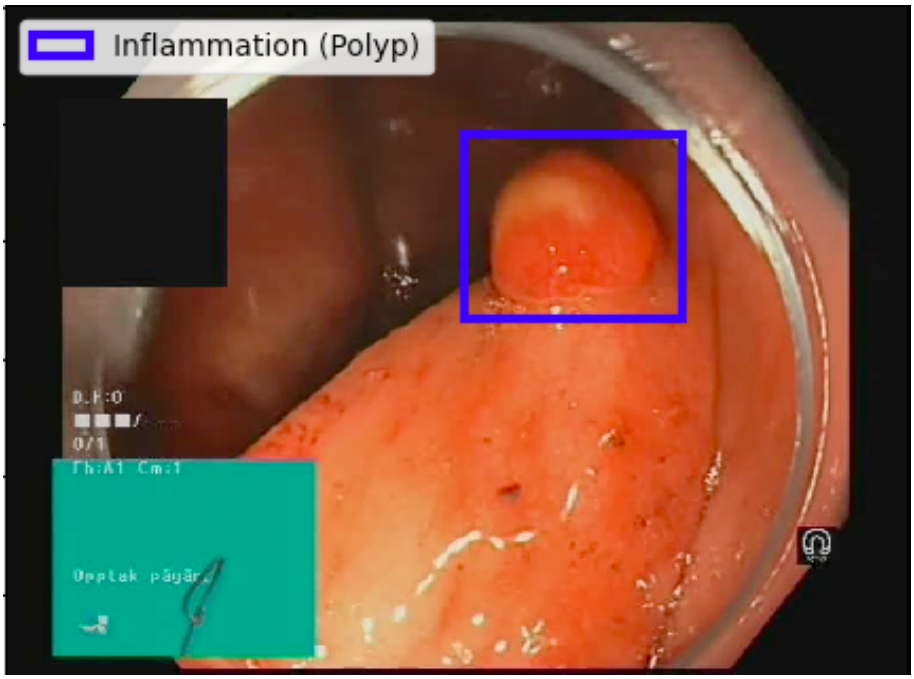
(a) An example of an image in the new dataset visualised with its corresponding bounding box and label (Inflammation (Biopsy)).



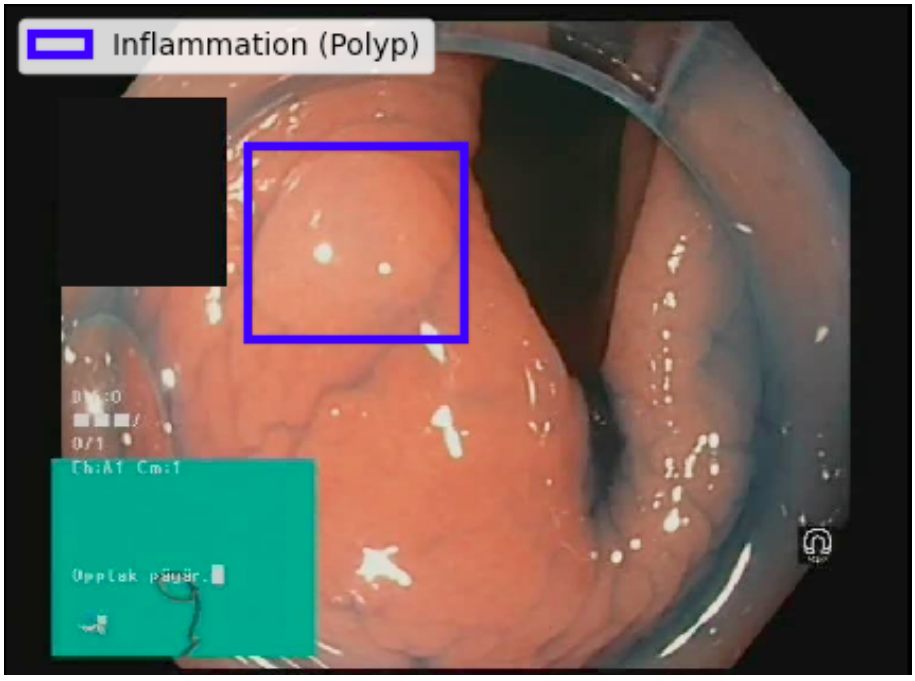
(b) Another example of an image in the new dataset visualised with its corresponding bounding box and label (Inflammation (Biopsy)).

Figure A.5: Two examples of images with Inflammation (Biopsy).

At last, in Figure A.6, two examples of images with Inflammation (Polyp) are visualised.



(a) An example of an image in the new dataset visualised with its corresponding bounding box and label (Inflammation (Polyp)).



(b) Another example of an image in the new dataset visualised with its corresponding bounding box and label (Inflammation (Polyp)).

Figure A.6: Two examples of images with Inflammation (Polyp).

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY