UiO : Department of Informatics
University of Oslo

Automatic Analysis of Endoscopic Videos

Deep Learning for Automatic Detection of Polyps in Endoscopic Videos

Torbjørn N. Høiland

Master's Thesis Spring 2017

Abstract

The human digestive system can be affected by many types of diseases. For example, three of the six most common cancer types (esophagus, stomach and colorectal) are located in the gastrointestinal tract. Colorectal cancer (CRC) is the third most common cancer in men and the second most common cancer in women worldwide, and Norway has one of the highest incidences of this cancer. Early detection is vital for the prognosis, level of treatment and survival.

EIR is a multimedia system with the main objective of supporting doctors in gastrointestinal tract disease detection, both as a live examination system and an offline system for VCE. However, the detection and automatic analysis subsystem within EIR today consists of two parts; the detection subsystem and the localisation subsystem.

Recent advances in machine learning, particularly deep learning, have provided excellent object detection models. This thesis explores the possibility of using a deep neural network at the base of the detection and automatic analysis subsystem in EIR, specifically by using You only look once (YOLO). YOLO is a state-of-the-art, real-time object detection system that was used together with the ASU Mayo Clinic polyp database to detect CRC precursors called polyps.

The YOLO system reaches a satisfactory detection accuracy, while still being able to process videos in real-time. The proposed system and EIR is compared using the standard metrics of recall, precision and F1-score. When compared, it is clear that the system still has room for improvement in regard to its precision.

Acknowledgments

First, I would like to thank my supervisors; Pål Halvorsen, Konstantin Pogorelov and Michael Riegler. Without their valuable feedback and input on my research and writing, this work would not have been possible.

Additionally, I would like to thank my family, especially my mom and sister, who has been a constant motivation and a helping hand with the writing of this work. Further, I would like to thank my good friend Simon, who was always there when I needed to vent about work (or anything really).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

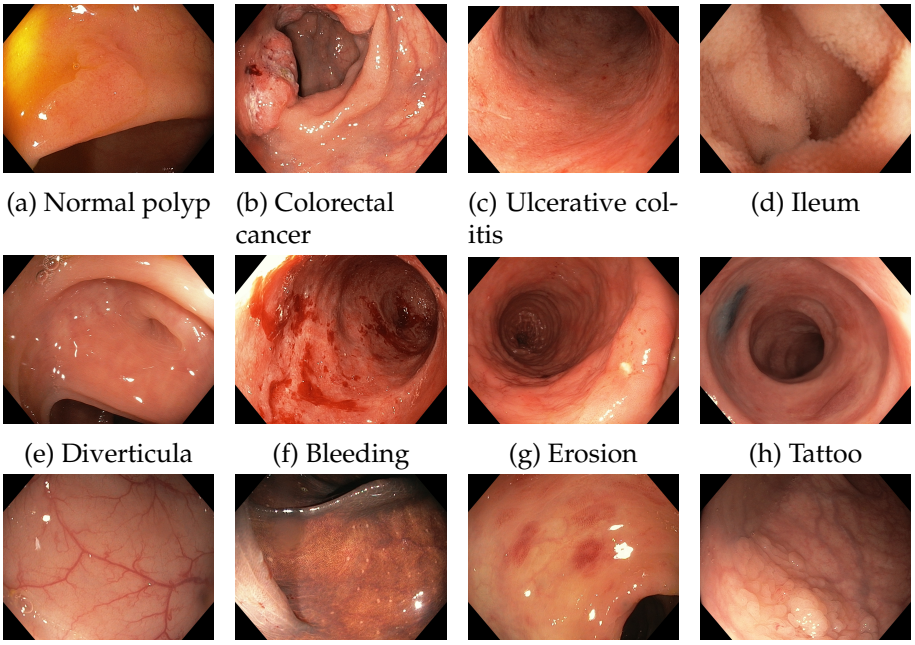
1.1 Background and Motivation

The human digestive system can be affected by many different types of diseases. For example, three of the six most common cancer types (esophagus, stomach, colorectal) are located in the gastrointestinal tract (GI tract) [44]. There are a lot of diseases that are common in the GI tract. Some visual examples can be seen in figure 1.1.

Colorectal cancer (CRC) is the third most common cancer in men and the second most common cancer in women worldwide [45]. Nearly 55% of the cases occur in the more developed regions and according to the Norwegian Cancer Registry [24], Norway has one of the highest incidences of CRC. The occurrence of this cancer type has more than doubled in the last 50 and so years. Studies also show that the 5-year survival rate of CRC ranges from 93% in stage 1 to 8% in stage 4 [33]. This makes early detection crucial for the prognosis, level of treatment and survival of the patients.

There are several ways of screening the GI tract, but population wide examinations are the most important tool for early detection. However, the current procedures have limitations regarding sensitivity, specificity and cost. The current recommended method for screening the colon is endoscopy, but it is a demanding procedure that requires a lot of time for medical personnel to complete. Furthermore, lesions are often missed due to the tiredness of the doctors or because a specific part of the colon was not accessible with an endoscope [22]. To help doctors by automatically analyzing videos during manual examinations and give live feedback, could prove to be quite helpful for the doctors. A way to accomplish this could be through the field of computer vision.

An alternative method for screening is to perform the examination using a wireless video capsule endoscope (VCE), which is swallowed



(a) Normal polyp (b) Colorectal cancer (c) Ulcerative colitis (d) Ileum

(e) Diverticula (f) Bleeding (g) Erosion (h) Tattoo

(i) Normal Vessel (j) Melanosis (k) Erythema (l) Advanced polyps

Figure 1.1: An inconclusive list of diseases that can be diagnosed using colonoscopy [47].

by a patient and then proceed with recording a video from the whole trip through the digestive system (GI tract). This, however, creates a tremendous amount of data (approx. 5-8 hours of video) and is difficult to scale for a population wide screening. A way of making these procedures better able to scale, could be through the field of computer vision.

Computer vision (CV) is the science of using computers to analyze and understand the content of images automatically. CV systems can automate tasks and remove the need for human labor. Advances in the machine learning field of Convolutional Neural Networks (CNNs) have improved the accuracy of CV systems considerably in recent years and many researchers believe that it has the potential to improve medical imaging [15]. In this work, it is investigated how adopting the use of these recent advances introduces the possibility of automatically analyzing and supporting medical experts both during procedures and on stored data from VCE examinations.

1.2 Problem Statement

Inspired by the return of deep learning these last few years the goal of this thesis is to explore a possible improvement for automatically diagnosing diseases in the GI tract. Building on the previous work from EIR [46], the focus will be to explore and use a neural network that can both: assist doctors with visual feedback during manual examinations and perform a fully automated screening of the GI tract using data from VCE examinations.

Being a medical assist tool there are strict requirements toward the accuracy of the system in order to avoid negative findings (overlooking a disease) as well as low resource consumption. To allow the system to assist doctors during examinations, it is necessary to introduce a real-time processing requirement (defined as processing at least 25-30 frames per second (FPS)) [47].

The specific problem statement for this thesis is: **can deep learning be used for automatic detection of diseases in the gastrointestinal tract?** The goal is to improve upon the work in EIR, which is presented in section 2.4.3, by introducing the use of modern techniques for object detection, specifically by introducing *deep learning* as a potential technology.

By introducing deep learning as a tool for automatic detection of diseases, we hope to further the research into a fast and efficient system for the detection and automatic analysis subsystem. To achieve this, the goals we hope to achieve with the system is as follows:

- High disease detection accuracy.
- Real-time processing for supporting medical experts during colonoscopies.
- Efficient processing to allow for screening with VCE.
- Being expandable to allow for detection of different diseases.

1.3 Limitations and Scope

Based on the research question in section 1.2, the scope of this thesis is researching a possible improvement to the detection and automatic analysis subsystem in EIR, which is presented in section 2.4.3. Now, because the size of the medical field is quite large, the focus for this investigation is the human gastrointestinal tract (GI tract) which can be affected by many types of diseases that are visually distinguishable. Further, it was decided to limit the disease detection system to polyps, because of two main challenges:

- The first challenge is the large number of possible diseases and their visually distinct appearances in the GI tract. See figure 1.1 for examples.
- The second challenge is the lack of publicly available data for different diseases, which makes evaluation and comparisons difficult between detection methods. At least for polyps there are some datasets available that can be used to evaluate the results for this work.

Limiting the detection system to polyp detection (in this thesis) is also because polyps are precursors to CRC, which makes them more interesting from a medical point of view. Because early removal of a polyp significantly decreases the chance for the patient to develop CRC, which makes early detection to be of clear value for survival.

1.4 Research Method

This thesis will be following the design paradigm described by the ACM Task Force in the report *Computing as a Discipline* [7]. This involves the design, implementation and evaluation of the object detection system where the goal is to achieve good detection accuracy of polyps in medical images.

The experiments with the object detection system follows an iterative approach, to allow for improvements and augmentations to be added after each evaluation. The results from the performance evaluation will then be used to adjust the parameters for the next experiment.

1.5 Main Contributions

In this thesis, we have shown that deep learning can be used for automatic detection of diseases in the GI tract. The preliminary results achieved, by using the YOLO object detection system, suggests that there is large potential for object detection systems within automatic analysis of medical imaging, specifically the detection of diseases in the GI tract. The systems both reach good detection accuracy, while still being within the real-time border of 30 FPS that was defined in section 1.2. Using either tiny-YOLO or YOLOv2 is possible, as tiny-YOLO runs detection fast (at 123 FPS), but has less accuracy, meaning that tiny-YOLO generates more false positives than the larger YOLOv2 network. This makes it possible to trade between accuracy and speed, by using either tiny-YOLO for its speed, or YOLOv2 for its precision (less false positives).

The main contribution in this thesis is the research and evaluation of a new, and potential improvement, to the detection and analysis subsystem in the multimedia system EIR. This system has proven able to detect polyps, with good accuracy, that can be seen in GI tract examination videos, either from a VCE or during manual procedures. It is, however, important to point out that the used dataset is limited in its size and that evaluations, and further experiments, on a larger amount of data is recommended.

1.6 Outline

The rest of this thesis is structured as follows:

Chapter 2 - Background: The background chapter presents the medical scenario and techniques for modern object detection. Further, it presents relevant research concerning the polyp detection use case and describes the previous work EIR, which this thesis builds on, and ends the chapter with a presentation of different systems that can be used for object detection.

Chapter 3 - Neural Network for Polyp Detection: This chapter introduces the goals for our system as well as giving a presentation of the object detection system, YOLO, that is used in the experiments. This is followed with how the system have been modified for our use case as well as a presentation of the dataset used for all the experiments.

Chapter 4 - Experiments & Discussion: This chapter begins with listing the exact setup used for conduction our experiments, before presenting the overall results that have been achieved. The chapter then continues with a presentation and discussion about the experiments and results, before ending with a general discussion about the results that has been achieved by the detection system.

Chapter 5 - Conclusion: The thesis is summarized and concluded before presenting some ideas for future work.

Chapter 2

Background

This thesis is about investigating a potential improvement to the detection and automatic analysis subsystem of EIR, which is presented in section 2.4.3.

This chapter will first provide necessary background information regarding the medical scenario that concerns the polyp detection use case. It will then continue with presenting modern techniques for object detection that can make it possible for computers to assist doctors during examinations and even act as a fully automated disease detection system. Further, the chapter will introduce the standard metrics that have been used for performance measuring of the detection system, before presenting research that is relevant towards the polyps detection use case. The chapter then continues with a presentation of EIR, the previous work of which this thesis builds upon, before ending with a presentation of different systems that can be used for object detection, leading up to the detection system that has been used in this thesis.

2.1 Medical Scenario

As stated in the introduction 1.1, CRC is the third most common cancer in men and second most common cancer in women worldwide [45]. The fact that Norway has one of the highest incidences of CRC [24] combined with the fact that lesions[1] are often missed due to the tiredness of the doctors [22], makes research into an automatic assist and disease detection system quite relevant as a research topic.

This section will further explain the medical scenario concerning procedures for examining and screening of the colon. This is necessary

[1] A lesion is any abnormal damage in the tissue of an organism, usually caused by disease (from wikipedia: https://en.wikipedia.org/wiki/Lesion)

as an introduction to the potential system for automatic detection of diseases in the GI tract.

The GI tract (see figure 2.1) can be affected by several diseases, where CRC is one of the major health issues worldwide. If CRC is detected at an early stage the prognosis is substantially improved [33], as said in section 1.1. Studies show that screening a large portion of the population improves the prognosis and also reduces the rate of CRC incidences [4, 24]. Therefore, the current European Union guidelines recommend screening for CRC for all people older than fifty years[2] [17].

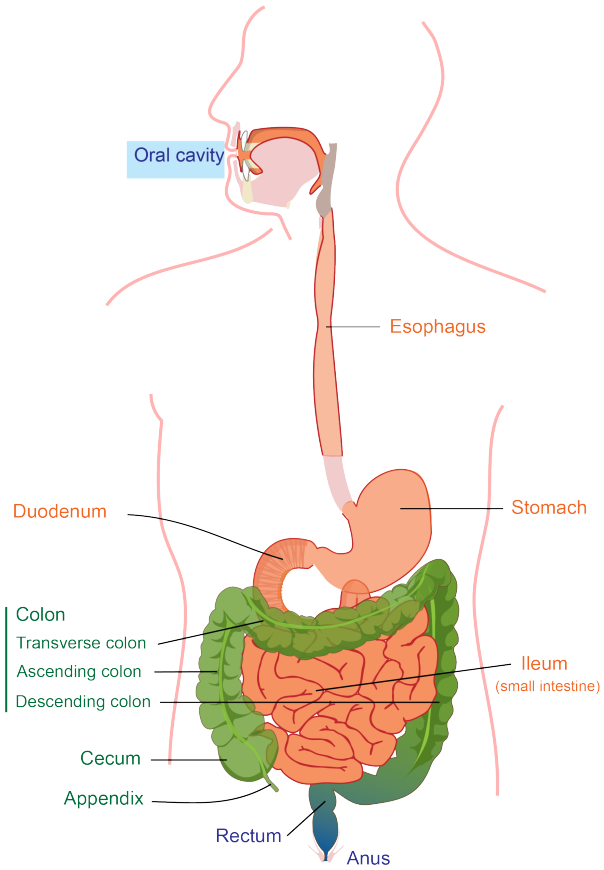


Figure 2.1: An overview of the gastrointestinal tract[3].

Polyps can be found in different parts of the body, like the GI tract, nose, bladder or stomach. Colon polyps are clumps of cells that form in the colon (seen in figure 1.1) and is a common precursor to CRC. Polyps

[2]The process of screening the GI tract is recommended to be performed on a regular basis: https://www.cancer.org/healthy/find-cancer-early/cancer-screening-guidelines/american-cancer-society-guidelines-for-the-early-detection-of-cancer.html

[3]The figure is derived from a diagram created by Mariana Ruiz and Joaquim Alves Gaspar. The original has been released into the public domain by the author: https://en.wikipedia.org/wiki/File:Digestive_system_diagram_edit.svg

typically, do not cause symptoms, particularly when they are small, but over time some may grow to become cancerous. There is no way to tell if a polyp will become cancerous, without it being analyzed in a laboratory, which acts as an incentive to remove any polyp once it is detected.

2.1.1 Colonoscopy

GI tract endoscopies are common medical examinations where the entire GI tract is visualized and examined to diagnose diseases. An endoscope is a long flexible tube (see figure 2.2) used to examine the interior of a patients body. The most common gold standard[4] for GI tract examinations are gastroscopy (entering via the mouth) and colonoscopy (entering via the anus).

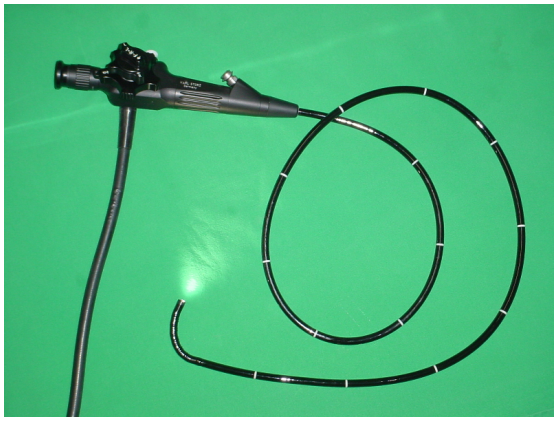


Figure 2.2: A standard endoscope[5].

Because some polyps *might* become cancerous, it makes early detection important for the prognosis, level of treatment and survival. The recommended method for detection is through endoscopy, but regular screening of the population is challenging due to several factors:

Invasive: Endoscopies are demanding invasive procedures and may lead to great discomfort for the patients.

Extensive training: There is extensive training required for the physicians to be able to perform the procedure.

Cost: The procedure is also quite expensive. For example, in the US, colonoscopy is the most expensive screening process, with annual

[4]The best available standard for an examination of a certain area in the body under reasonable conditions.

[5]Image is from Wikipedia: `https://en.wikipedia.org/wiki/Endoscopy`.

costs of 10 billion US dollars[6].

Time: A colonoscopy requires about one medical-doctor-hour and two nurse-hours per examination.

It is apparent that scaling colonoscopy procedures to a population-wide process requires significant resources, incurs large costs and, while the use of endoscopy is the preferred method for detection, the procedure is not ideal. On average 20% of polyps are missed or not removed completely, so the risk of developing colorectal cancer largely depends on the physician performing the procedure [22].

2.1.2 Virtual Colonoscopy

Another procedure that is used to screen the colon is virtual colonoscopy (see figure 2.3), which is a minimally invasive examination. During this screening process a CT scan produces many cross-sectional images of the patients abdominal organs. The images, which can be two- and three-dimensional images, are then combined and digitally manipulated to give physicians a detailed view inside the colon. One disadvantage of this procedure is that the detail provided is reduced, as opposed to conventional endoscope, and this means that polyps smaller than 2-10 millimeters in diameter may not show in the produced images. Another disadvantage to this screening method is that the patient is exposed to a significant amount of radiation during the scan.

Figure 2.3: CT machine for virtual colonoscopy[7].

[6]Numbers from the NY times: http://www.nytimes.com/2013/06/02/health/colonoscopies-explain-why-us-leads-the-world-in-health-expenditures.html?pagewanted=all&_r=0.

[7]Image is from wikipedia: https://en.wikipedia.org/wiki/CT_scan.

2.1.3 Wireless Video Capsule Endoscopy

Medical screenings used to identify undiagnosed diseases in large populations have some known issues, like too many false positives, invasive screening procedures and high costs [22]. However, the benefits of screening can outweigh the disadvantages.

A solution that can provide a more efficient and large scale screening, is the use of camera pills, as shown in figure 2.4. Video Capsule Endoscopy (VCE) is yet another way of screening the gastrointestinal tract of a patient, and it has the potential to overcome several of the limitations and disadvantages arising from the previously discussed procedures in sections 2.1.1 and 2.1.2.

Screening a patient with VCEs is done using a small capsule, in essence, a camera masked as a pill. When the capsule is swallowed, it moves through the entire gastrointestinal tract, continuously recording and transmitting captured images to a receiver worn on a belt around the patients waist.

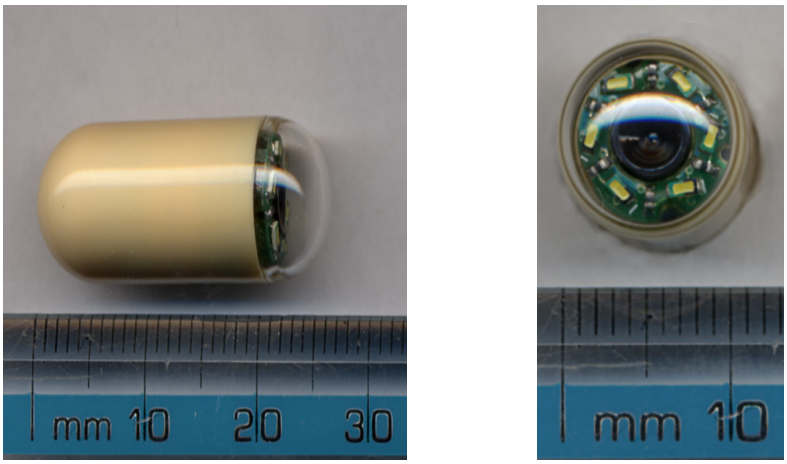


Figure 2.4: A look on a video capsule endoscope[8].

VCE has a considerable potential as a tool for screening the gastrointestinal tract; however, a drawback is the large amount of data, about 5-8 hours [9] of video, captured with each procedure. This is an inordinate amount of time for a medical professional to invest in a single procedure, and to make VCE meet the needed efficiency in health care it is necessary to automate as much of this process as possible.

In the literature, current computer aided diagnosis (CAD) techniques generally uses feature extraction of color, shape and texture combined with machine learning classifiers to detect colon polyps [16, 18, 19].

[8]Both images have been released into the public domain:
https://en.wikipedia.org/wiki/File:CapsuleEndoscope.jpg,
https://en.wikipedia.org/wiki/File:CapsuleEndoscopeEnd.jpg.

The main difficulty of the feature extraction methods is due to several factors, e.g. a variance in illumination, the different shapes of polyps or the blurring of frames due to the movement of the capsule [18, 58].

A potential solution, could be training a neural network to do the feature extraction and making it possible to avoid both the manual work and create better descriptors for detecting polyps. With a neural network, it is also possible to expand the detection to more irregularities and diseases, e.g., bleeding, by expanding the training data. But first, it would be prudent to have a short introduction to a subfield of artificial intelligence that is known as deep learning.

2.2 Artificial Intelligence

Today, artificial intelligence (AI) is a highly active field of research and practical applications. It is widely used to automate tasks, understanding speech and images, make diagnosis in medicine and support basic scientific research. The true challenge of AI is solving tasks that are easy for humans to perform, but difficult to explain, tasks that we solve intuitively, like recognizing speech or faces in images. A solution to this, as stated on page 1 in the introduction chapter from the deep learning book [14]:

> *"[..] the solution to this is to allow computers to learn from experience and understand the world in terms of a hierarchy of concepts, with each concept defined in terms of its relation to simpler concepts".*

Drawing a graph that displays how these concepts are relative to each other, the graph is deep with many layers, which is why this approach to AI is called *deep learning*.

2.2.1 Deep learning

Deep learning is an approach to AI, specifically, it is an approach to machine learning that has drawn from our knowledge of the human brain. In the last few years there has been a tremendous growth in its usefulness and popularity, due to larger and more powerful computers and techniques for training deeper networks.

An artificial neural network is built up of multiple layers of nodes, or neurons, where each node has connections to nodes from the previous layer. As seen in figure 2.5, the first layer is referred to as the input

[9] This is figure 1.2 from the deep learning book [14]. It can be found on page 6 in the introduction.

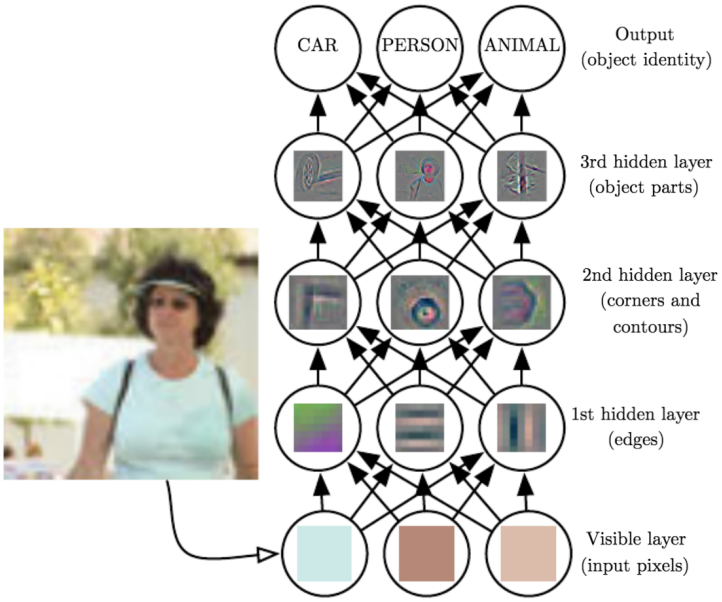


Figure 2.5: Deep learning overview[9].

layer and the last layer is referred to as the output layer. The layers in between, are the hidden layers, as these are not directly observable when running evaluation. Note that when a neural network has multiple hidden layers it is often referred to as a deep neural network (DNN).

Figure 2.5, gives a view of how deep learning can interpret the world with a hierarchy of concepts. As one can see, it would be difficult for a computer to draw any meaning from the raw input data. By using deep learning, the computer can separate the complicated task into simpler ones. The first hidden layer can find edge features from the image by comparing with neighboring pixels and then the second layer combines these edge features to find contours and corners in the image, which in turn is used to describe parts of the object in the image.

The current methods for deep learning, provide a powerful framework for supervised learning[10] and making networks deeper, by adding more layers and nodes per layer, a deep network can represent increasingly complex functions. Tasks that are easy for a person to do rapidly, like recognizing objects in a video, can be accomplished using deep learning, as long as the models and datasets with labeled training examples are sufficiently large.

[10]Refer to section 2.5.1 for more on supervised learning.

2.2.2 Convolutional Neural Networks

A neural network is a system of interconnected artificial neurons that can exchange information between each other. The connections between the neurons have numeric weights that are updated during the training process, in such a way that a properly trained network will respond correctly when presented with a pattern or image to recognize. One can think of these neurons as simple feature detectors, where each layer of neurons responds to different combinations of inputs from previous layers (see figure 2.5). The layers are stacked in such a way that the first layer detects a set of primitive features, the second layer detects features of these features, and so on.

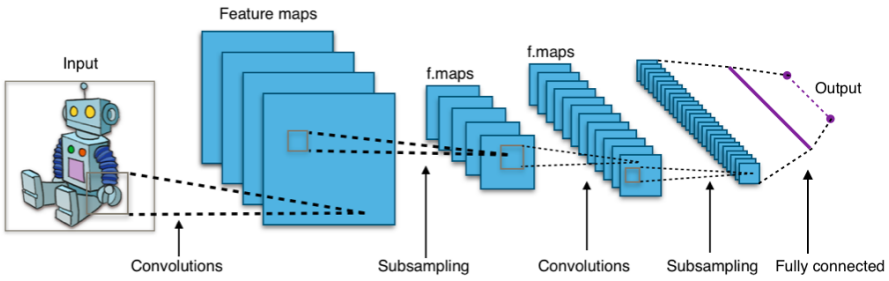


Figure 2.6: A typical CNN architecture[11].

The design of Convolutional Neural Networks (CNNs) are biologically inspired by the behavior of a visual cortex. The cortex contains several cells that are responsible for detecting light in small, overlapping sub-regions, which are called receptive fields. These cells act as filters over the input space, where the more complex cells have larger receptive fields. The convolution layer of a CNN performs the function that is performed by cells in the visual cortex [51, 64].

A CNN is a special implementation of the neural network and is typically used for pattern- and image-recognition. A CNN is built up of one or more convolution layers and often with a pooling layer, which is then followed by one or more fully connected layers as in a standard neural network. As we can see from figure 2.6, each feature of a layer receives inputs from a small neighborhood of features (called a local receptive field) in the previous layer.

When training a CNN on different tasks with images as input, a lot of similar features are found in the first few layers no matter what the network is being trained to detect. From the figure 2.5, one can see that for the first layers the features that are extracted are quite simple, for example, *edges* are a basic feature found in any image. Based on this, it seems wasteful to train new networks completely from scratch each

[11]The image has been released into the public domain by the author:https://en.wikipedia.org/wiki/File:Typical_cnn.png.

time. Studies actually show that one can save a lot of training time and shows that it is possible to get good results with smaller training dataset, by using a pre-trained model instead of random initial weights [13, 55].

2.3 Performance Measuring

To evaluate how well different object detection systems perform on a set of test data, a score of the accuracy for each system is useful. The metrics used to evaluate the system presented in this thesis are: *precision*, *recall* and *F1 score*.

Precision denotes the number of predictions, made by the detection system, that were correct. It is computed by true positives (*tp*, correctly classified as positive) divided by *tp* plus false positives (*fp*, falsely classified as positive class).

$$Precision = \frac{tp}{tp + fp}$$

Recall (aka. sensitivity) denotes the number of ground truth bounding boxes that were accurately predicted. It is computed by *tp* divided by *tp* plus false negatives (*fn*, falsely classified as negative class).

$$Recall = \frac{tp}{tp + fn}$$

In an optimal case, both *precision* and *recall*, are high. Usually these values are in such a relation with each other, so that a high precision leads to lower recall and vice versa [39]. A way to calculate the prediction quality of a network that considers both measures is the F1-score. F1-score is basically the probability that the networks predictions are correct. It is an average of both measures, more specifically, it is the harmonic mean between precision and recall.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

A *tp* is found when there exists a certain amount of overlap between a predicted bounding box, made by the network, and a ground truth bounding box. This overlap is defined by the Intersection-Over-Union (IOU) of the prediction and ground truth box, the intersection of the two boxes divided by the union of their areas. A typical threshold value signifying a correct detection is an IOU of 50%.

Predictions made by the network are not absolute. Each prediction has its own probability or confidence score that tells us how certain the

prediction is. By choosing a minimum confidence value, often referred to as a threshold, the total number of predictions will be reduced. This will increase the precision of the network because of the lower number of false detections. However, a higher threshold also means that recall decreases, which makes selecting a good threshold value into a matter of finding a balance between good precision and high recall.

2.4 Polyp Detection

Based on the discussions from sections 1.1 and 2.1 there is a need to develop a detection system for GI tract examinations, that can both act as a live CAD tool during procedures and a scalable screening system for VCE videos. Now, a system that aims to analyze the entire GI tract needs to fulfill some specific requirements. It needs to be able to process a large amount of data efficiently, while still being practically applicable so that it can support doctors during examination procedures or help analyzing VCE videos.

As said in section 1.3, detection of diseases in the GI tract are mostly focused on polyps due to lack of available data and the fact that polyps are precursors to CRC. Automatic analysis of video from colonoscopies has attracted research attention for a long time, with several studies published [59, 61, 62].

In the following sections, related work that is relevant for this thesis will be presented.

2.4.1 Machine learning approaches

Machine learning algorithms can be separated into supervised and unsupervised learning (see section 2.5.1) and two-class or multi-class algorithms. Two-class algorithms are able to predict if an image contains one specific object, e.g., if the image contains a dog or not. The multi-class classifiers are not limited to a number of classes and can find multiple objects in an image [11, 30]. An example of how a multi-class algorithm could work can be seen in figure 2.8b.

Concerning the classification of multimedia data, there exists a lot of alternatives. Here is a presentation of the four most popular machine learning approaches:

Support Vector Machines (SVM): Operates with training data that has two labeled classes. They can classify future data with similar features into their corresponding classes. This is done by mapping data as points in a space, so that the two classes are clearly separated [11, 53].

Instance-based algorithms: These algorithms learn from known and already labeled training data. This works by comparing new data with training data to make predictions [1, 11, 30].

Clustering: This is a well known machine learning method used in unsupervised learning. This approach is used to cluster data into different groups based on similarities [11, 30].

Deep Learning: This is based on neural networks and is a machine learning approach that has drawn from our knowledge of the human brain. More about deep learning approaches in section 2.4.2 and an introduction to what deep learning is can be found in section 2.2.

The first three approaches, SVM, instance-based algorithms and clustering are well researched approaches and are considered in the category of traditional machine learning. Deep learning, on the other hand, is a rather new approach that has become popular these last few years.

For automatically classifying polyps in endoscopic imaging data most approaches rely on SVMs or instance-based two-class classifiers. The specific *features*, from the image, vary depending on each approach. These features can be physical dimensions, greyscale intensity values, gradient orientation, color information or texture and are used as input into the classifier. For automatic analysis, one can generalize these methods into two different approaches: geometrical analysis and machine learning.

Table 2.1 gives a summary of approaches for polyp detection that are most relevant. The last row of the table shows the performance of EIR, which will be presented in section 2.4.3.

Mamonov et al. [29] presented a method called binary classification with pre-selection. It is a binary classifier for detecting polyps in the colon, focusing on reducing the number of frames that need to be manually inspected by medical doctors. As can be seen from the table 2.1, the algorithm reaches a recall rate of 47% with regards to single input frames.

Hwang et al. [20] is a similar approach and also focuses on polyp shapes, particularly ellipses. This method first segments a frame into different regions by using a watershed-based image segmentation algorithm, which is based on observations that polyps are spherical or hemispherical geometric elevations. When finding a potential polyp, the subsequent frames are examined for the same characteristics. This enables the system to apply a threshold for the number of subsequent frames after the detection, to reduce the number of false positives.

Wang et al. [60] presents the most complete system in the polyp

Publication/System	Detection Type	Recall/Sensitivity	Precision	Specificity	Accuracy	FPS	Dataset Size
Wang et al. [60]	polyp/edge, texture	97.70%[a]	N/A	N/A	95.70%	10	1.8m frames
Tajbakhsh et al. [55]	polyp/shape, color, texture	50%	-	-	-	-	35,000 images
Park et al. [34]	polyp/shape, color, texture	82.80%	65.80%	-	-	-	62 images
Wang et al. [63]	polyp/shape, color, texture	81.4%	-	-	-	0.14	1,513 images
Mamonov et al. [29]	polyp/shape	47%	-	90%	-	-	18,738 frames
Hwang et al. [20]	polyp/shape	96%	83%	-	-	15	8,621 frames
Li and Meng [27]	tumor/textural pattern	88.6%	-	96.2%	92.4%	-	-
Zhou et al. [66]	polyp/intensity	75%	-	95.92%	90.77%	-	-
Alexandre et al. [2]	polyp/color pattern	93.69%	-	76.89%	-	-	35 images
Kang et al. [23]	polyp/shape, color	-	-	-	-	1	-
Cheng et al. [5]	polyp/texture, color	86.2%	-	-	-	0.076	74 images
Ameling et al. [3]	polyp/texture	AUC=95%[b]	-	-	-	-	1,736 images
Nawarathna et al. [31]	abnormalities/features	92%	-	91.8%	-	-	-
EIR	abnormalities/30 features	98.50%	93.88%	72.49%	87.70%	~ 75[c]	18,781 frames

[a]The sensitivity is based on the number of detected polyps, other papers use per frame detection.

[b]Reported only area under the curve (AUC) instead of sensitivity.

[c]Detection and localization performed together for full HD videos. Detection performance alone is around 300 FPS and for localization around 100 FPS.

Table 2.1: Performance comparison of polyp detection approaches discussed in this chapter. Not all performance measurements are available for all methods. Nevertheless, including all the available information gives an idea about each methods performance.

detection field. It is called Polyp-Alert and reaches near real-time feedback for colonoscopy examinations. It employs their previous edge-cross-section visual features with a rule-based classifier to detect the edge of a polyp. They also track the detected polyp edge (or edges), so that they can group a sequence of images that covers the same polyp (or polyps). The paper report a recall of 97.7% on the data, which is 52 videos recorded during colonoscopies using different endoscopes. The fact that the testing data they use is not publicly available makes an exact performance comparison impossible.

Another approach, which is not limited to only detecting polyps, is presented by Nawarathna et al. [31]. The proposed method uses a texton histogram of an image block as features to determine if there exists an abnormality in the image, achieving 92% recall and 91.8% specificity on VCE images and 91% recall and 90.8% specificity on colonoscopy images.

2.4.2 Deep learning approaches

Deep learning algorithms are based on neural networks and use recently developed techniques to train their models. A more detailed dive into deep learning can be found in section 2.2. The new training techniques used in deep learning were mainly made possible due to the emergence of GPU computing, which allows for completion of training within a reasonable amount of time [26]. The main disadvantages of applying deep learning, is the time to convergence during training (long time to train), the fact that classifications made by the network are hard

to explain (acts as a black box system) and the network is very data driven [11, 26, 30]. On the other hand, deep learning can work very well for multi-class classification [11, 26, 30].

For similar problems, like detecting breast cancer [57], lung cancer [6] or polyp detection [55], deep learning is already very relevant. However, training is very complicated and time-consuming. The fact that it is difficult to evaluate the specific classifications that a network performs can lead to serious problems in the medical field [32]. Additionally, one of the largest challenges with deep learning is the large quantities of data required to train a robust detection system. Data collection is really difficult due to legal and ethical issues as well as the very high workload of doctors which makes it difficult to get ground truth [12] data.

As one can see from table 2.1, recent approaches for polyp detection, using neural networks, can achieve interesting results with a relatively small labeled dataset. Tajbakhsh et al. [55] presents a polyp detection method that is based on a 3-way image presentation and CNNs. The method consists of learning polyp features, color, shape and texture, in multiple scales. Given a polyp candidate, a set of CNNs specialized in each of these features are applied and their results are combined to either accept or decline the candidate. This method has a detection performance of 0.002 false positives[13] per input frame at 50% recall.

Park et al. [34] presents another method that also adopts the use of a CNN as their main tool. This approach focuses on polyp shape detection by using scale-invariant learning of hierarchical features. This method achieves a recall of about 83% with 66% precision on a total of 62 images that contains 64 polyps.

In the following section, the work that is the basis for this thesis, will be presented.

2.4.3 Previous Work - EIR

As said in section 1.2, this thesis builds on the work of EIR [35, 38, 46, 47, 48, 49, 50], which is a multimedia system that aims to detect diseases in the GI tract.

The EIR system consists of three subsystems:

Annotation Subsystem: This subsystem has the task of collecting training data for the detection and automatic analysis subsystem. This data can only be collected with the help of medical experts.

[12] Ground truth, or labels, which tells us where the object is in the image. E.g. binary ground truth images seen in figure 3.4.

[13] False positives are explained in section 2.3.

Detection and Automatic Analysis Subsystem: This subsystem has the purpose of automatically detecting, analyzing and localizing endoscopic findings in the GI tract for standard colonoscopies and VCEs. This subsystem is designed in a modular way to simplify future improvements and widen the field of different diseases that it can detect. As it is now, this subsystem consists of two parts:

1. The *detection* part that detects irregularities in video frames and images. This means it actually acts as *classification* (see section 2.5.1).
2. The *localization* part that finds the exact location of the irregularities classified in the first part.

Both of these parts, detection and localization, combined acts as the object detection system that *finds* objects in videos or images (see section 2.5.1).

Visualization Subsystem: This subsystem has the task of providing the results from the automatic detection and analysis subsystem to the medical experts that are supposed to use the output for CAD.

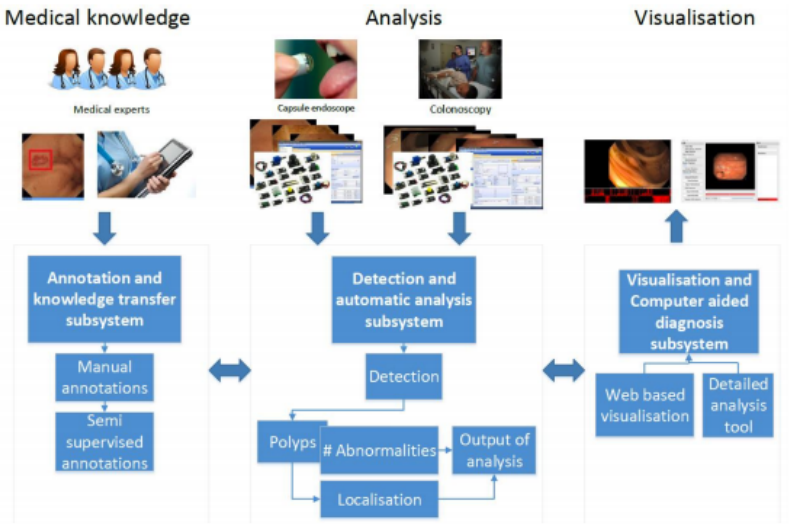


Figure 2.7: A complete overview of the EIR system[14].

As the focus in this thesis, specified in section 1.2, is the automatic detection of diseases (read *polyps*) in the GI tract, this means that the **detection and automatic analysis subsystem** is the subsystem of EIR that is to be further investigated.

The detection and automatic analysis subsystem compares and determines abnormalities in the given video frames or images by using global image features. To achieve this, the Lire [28] open source library for

[14]Figure can be found in [46] as "Figure 3.1".

content-based image retrieval is used. Lire provides a comprehensive set of algorithms to extract different types of global image features. Once the global image features have been extracted and stored in an index, which is performed by the *detection* part of the subsystem, this index is then used as input for the *localization* part of the subsystem.

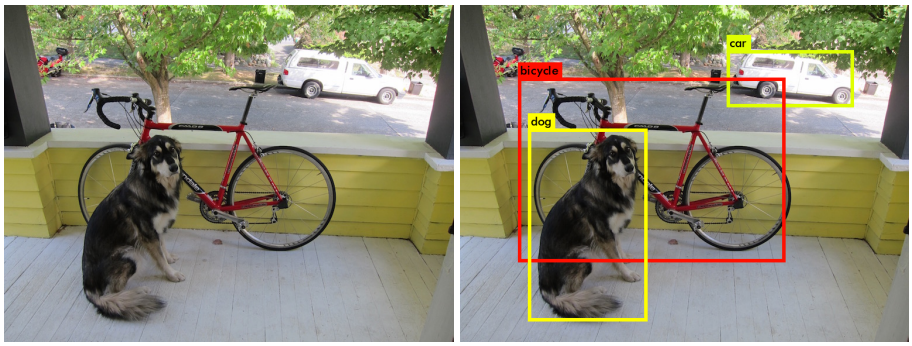
The localization part of the subsystem then has the task of finding the exact location of the detected irregularities. The images are filtered by a sequence of intra-frame filters, to allow for better localization that uses local image features.

2.5 Object Detection Systems

This section introduces the idea of object detection, before presenting a few different CNN-based object detection systems. The section ends with a presentation of the system that has been used in this thesis, YOLO.

2.5.1 Object detection

Object detection is a subfield of computer vision that deals with identifying and locating specific objects within an image or video. For instance, given the image in figure 2.8a, the goal would be to predict the bounding boxes for the objects as shown in figure 2.8b.



(a) Example image.

(b) Exmaple image with bounding boxes applied.

Figure 2.8: Example image with and without bounding boxes around detected objects[15].

Object detection differs from the similar challenge of *image classification* where the goal is to describe the image with labels. In the case of figure 2.8a this could be *dog*, *bike* and *car*.

[15]Image is from the YOLO homepage: https://pjreddie.com/darknet/yolo/.

Machine learning[16] has become an important part of object detection research, as the problem is exceedingly complex. Machine learning is the study (and construction) of algorithms that can learn from examples, which is especially useful for high dimensional and complex problems, where it is easier to get example data than to program a direct solution.

Machine learning is often divided into unsupervised and supervised learning. Unsupervised learning focuses on finding structure in *unlabeled*[17] data. Supervised learning, on the other hand, is given a *labeled* set of data that has input-output pairs. The goal of this is to be able to predict the right output from new data that was not a part of the training data. In object detection, the training data would then be pairs of input images and output bounding boxes (labels) for the specific objects, with the goal being to predict bounding boxes on previously unseen images.

However, measuring the performance of object detection has to be done on a separate dataset, often called the validation dataset, because of a common problem within machine learning that is known as overfitting. Overfitting is what occurs when a trained model learns features (or noise) that is specific to the training data, but not relevant when examining new, previously unseen data. This lays the groundwork for one of the main challenges within machine learning, which is to generalize the model (or learned features) well enough, so that it can recognize the same object in new data.

In the following sections, we will introduce a few different CNN-based object detection systems leading up to YOLO, the system that has been used in this thesis.

2.5.2 R-CNN

Inspired by the work of Krizhevsky et al. [25], on the CNN-based classifier that outperformed earlier classification techniques, Girshik et al. [13] proposed an object detection system that combined a CNN with region proposals, called R-CNN.

The figure 2.9, shows that the R-CNN detection system operates with three steps. The first steps generates category-independent region proposals which acts as proposals that define a set of candidate detection possibilities for the detector. These regions then act as input to a large CNN that extracts a 4096-dimensional feature vector. The

[16] More on Machine learning can be found here: https://en.wikipedia.org/wiki/Machine_learning.

[17] A deeper explanation can be found here: http://stackoverflow.com/questions/19170603/what-is-the-difference-between-labeled-and-unlabeled-data.

[18] Image is from Girshik et al. [13], the R-CNN paper.

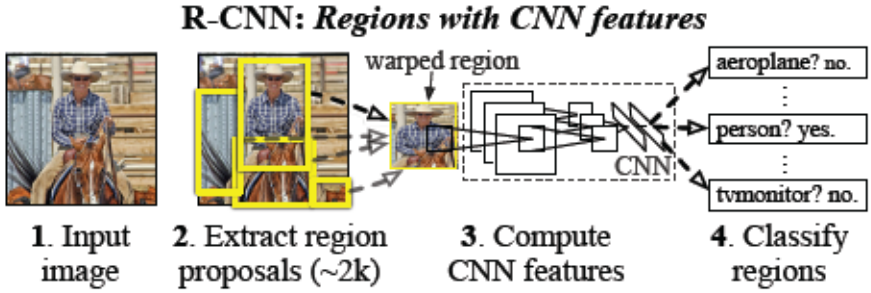


Figure 2.9: R-CNN overview[18].

final step is a set of class specific SVMs that score each extracted feature vector for a specific class. Based on all the scored regions, for all the classes, only the highest scoring classes for each overlapping region are kept.

The R-CNN algorithm is effective because of the fact that the CNN parameters are shared across all categories. This means that step one, from figure 2.9, only has to be executed once for all classes. The evaluation of the network is also only done once per region (from step one) which results in lower computational costs.

R-CNN still has drawbacks. The training pipeline, consisting of fine-tuning a CNN, fitting SVMs to the feature vectors and learning bounding box regressors, is complex, which renders training to be slow (up to 2.5 GPU days[19] on 5000 images [12]). One of the major bottlenecks of the R-CNN system in test-time is the fact that it performs one CNN forward pass for each object proposal.

2.5.3 Fast and Faster R-CNN

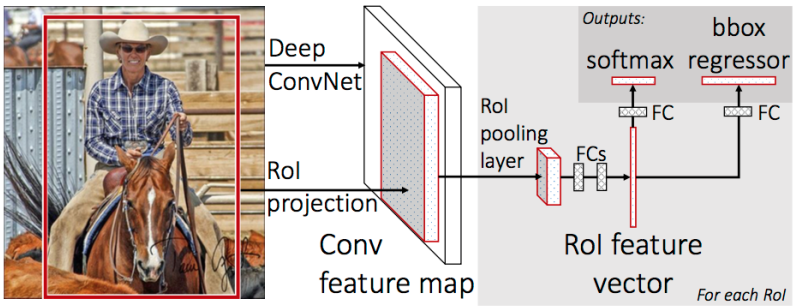


Figure 2.10: Fast R-CNN architecture[20].

[19]One GPU day is 24 hours on a single GPU.
[20]Image is from the Fast R-CNN paper [12].

Fast R-CNN was proposed by Ross Girshik [12], to improve the training and testing speed of R-CNN. As you can see in figure 2.10, the Fast R-CNN network takes an entire image as input with a set of object proposals. The image is then processed through several convolutional and maxpooling layers to produce a convolutional feature map. After this, a region of interest (RoI) pooling layer, extracts a feature vector from the feature map, for each of the object proposals. Every feature vector is then processed by fully connected (FCs) layers that branches into two output layers: The softmax output layer that produces a probability estimate for each object class, and a bounding box regressor returning refined bounding-box positions for each class.

Fast R-CNN still relies on slow algorithms, like selective search, for generating initial object proposals. As said in [43], selective search usually uses 2 seconds per image in a CPU implementation, limiting the frame rate to under 0.5 FPS, which lowers the benefits of speeding up the network further.

Faster R-CNN [43] presents an approach that improves on the way region proposals are computed. The approach introduces a Region Proposal Network (RPN), which acts as a kind of fully convolutional network and can be trained specifically for the task of generating region proposals. The effective processing time for the RPN is only 10 milliseconds per image. The total pipeline for Faster R-CNN reaches 5 FPS on the very deep VGG-16[21] model.

As stated in section 1.2, the detection system to be used for polyp detection needs to be able to run detections in real-time, which means that the object detection in R-CNN, with VGG16, that takes 47 seconds per image, is too slow. The Faster R-CNN model reaches 5 FPS on the same VGG16 model and then, with the smaller ZFnet [65], the FPS increases to 17. As our definition of real-time was to process 25-30 FPS, from section 1.2, the FPS reached is too slow for our use case. For this reason, the object detection system that has been used in this thesis is You Only Look Once, or YOLO for short.

2.5.4 YOLO

Recent approaches for detection systems, like R-CNN 2.5.2, uses regional proposal methods to generate potential bounding boxes for an image and then run a classifier on these boxes. The systems then have post-processing methods that refine the bounding boxes, eliminates duplicates and evaluates the boxes against other boxes in the scene [13]. These complex pipelines are slow and hard to optimize, because every component is trained separately.

[21] VGG16 is a pre-trained model [52].
[22] The image is from the YOLO paper [41].

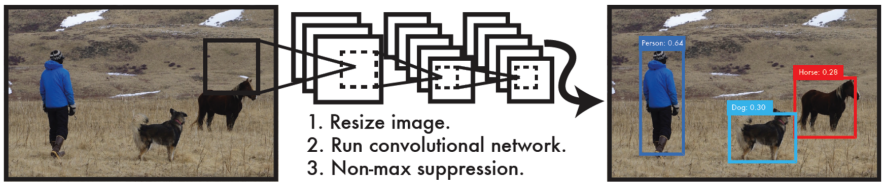


Figure 2.11: The YOLO detection system[22]as presented in [41]. All processing is accomplished by a single deep CNN.

YOLO, published in 2015 by Redmon et al. [41], is based on Darknet, which is an open source neural network framework [40]. It uses a different approach than the classifier-based systems. Its predictions are based on the entire image and made using a single network evaluation, as shown in figure 2.11. From the paper [41]:

> *"We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Using our system, you only look once at an image to predict what objects are present and where they are."*

YOLO works by dividing the input image into an $S \times S$ grid. When the center of an object falls within a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes with a corresponding confidence score, which reflects how confident the model is that the bounding box contains an object and how accurate the box is.

As discussed earlier, the detection system needs to operate in real-time. The reported detection frame rate of YOLO is 45 FPS (on a Titan X GPU), which means it can process video in real-time, as was listed as a requirement in section 1.2.

In late 2016, Redmon et al. [42], introduced YOLOv2. It focuses on improving the recall and localization error from the previous YOLO version. Instead of increasing the size of the network they simplify it and make representations easier to learn. To improve YOLOs performance several methods have been added:

Batch Normalization: Leads to lower time requirement for training while eliminating the need for other forms of regularization [21].

High Resolution Classifier: Fine tuning of the classification network is done at full 448×448 resolution for 10 epochs on ImageNet (pre-trained model/weights).

Convolutional with Anchor Boxes: Removes the fully connected layers from previous YOLO model and uses anchor boxes to predict bounding boxes. This creates two issues when using it in YOLO:

1. **Dimension Clusters:** The first issue is that the box dimensions are handpicked. Instead of choosing this by hand, k-means clustering is used on the training set bounding boxes to help the network automatically find good dimensions for the bounding box. This is done to find boxes that give a better IOU[23] score.

2. **Direct Location Prediction:** The second issue is model instability, especially during early iterations. Using random initialization, the model takes a long time to stabilize to predicting sensible offsets for the bounding boxes. Instead YOLO predicts location coordinates relative to the location of the grid cell.

Fine-Grained Features: Helps YOLO with localizing smaller objects by adding a pass-through layer that increases the resolution of the feature map.

Multi-Scale Training: Because the model only uses convolution and pooling layers it can be rescaled on the fly. By resizing the network every few iterations, the model becomes robust for images in different sizes.

Considering these improvements to the YOLO model and that the newer version also reported running detection in real-time, it was decided to use YOLOv2 for the experiments in this thesis. Also notable is the existence of a faster and smaller network architecture called tiny-YOLO. This network is built to be lightweight and fast, but has less detection accuracy than the larger YOLOv2 model. Tiny-YOLO has a reported FPS of more than 200 when running on a GPU[24].

2.5.5 Tiny-YOLO vs. YOLOv2

Figure 2.12: The YOLO Logo[25].

As mentioned at the end of section 2.5.4, the experiments in this thesis are being conducted with YOLO, using two pre-defined network structures, tiny-YOLO and YOLOv2. You have the large network,

[23] See section 2.3.
[24] From the YOLO website: https://pjreddie.com/darknet/yolo/#tiny.
[25] Logo is from the YOLO website: https://pjreddie.com/darknet/yolo/.

YOLOv2, with 22 convolutional layers and then tiny-YOLO, with its 9 convolutional layers. The Tiny-YOLO model is designed to be a trade-off between detection accuracy and speed, while also requiring less memory to run.

From the github of AlexeyAB[26], which actually provides a port of YOLO for windows, you can find that the pre-trained models require different amounts of GPU memory. The small network, tiny-YOLO, requires 1 GB of GPU memory, while the full sized YOLOv2, requires up to 4 GB GPU memory[27].

Using two networks for the experiments in this thesis was to investigate whether there is a correlation between the resulting detection performance when running on the same data. The networks are trained and evaluated using the same dataset (separate dataset for training and evaluation, of course).

Using tiny-YOLO is interesting for several reasons. First, the speed in which it is designed for, makes it very interesting for running detection in real-time. Secondly, the reduced training time means it allows for running more experiments in the same amount of time as YOLOv2. In our experience, the YOLOv2 network uses about twice as long for training as tiny-YOLO[28]. Thirdly, the lower memory usage makes it possible to run detections on cheaper (and smaller) hardware.

Comparing the detection performance between the two differently sized networks can also tell us whether tiny-YOLO can act as a proxy for how the YOLOv2 network will perform; that is, if training on one dataset is better for tiny-YOLO is it likely to be better for the larger YOLOv2 network as well? If this is true, one can use tiny-YOLO to test for performance on datasets, to allow running the experiments at a faster rate, and then apply the final training regime to the YOLOv2 network, with its reported higher detection accuracy.

2.6 Summary

In summary, actual computer-aided diagnostic systems for the GI tract do not provide real-time performance, with sufficient detection and localization accuracy required when operating within the medical field. Most of the approaches in the GI tract use case, uses evaluation approaches that are often based on small amounts of data, or data that are not publicly available. Deep learning approaches seem to be a hot topic in the medical imaging processing field, even though they

[26] GitHub of AlexeyAB: https://github.com/AlexeyAB/darknet.

[27] The amount of memory usage depends on the training scheme, e.g. when having multi-scale training enabled the YOLOv2 network actually used 7980MB of memory.

[28] Training with the hardware specified in section 4.1.1.

require a large amount of data for training. Nevertheless, deep learning has a large potential for automating disease detection, which makes more research into these modern object detection systems, in regard to medical image processing, seem necessary.

In the next chapters, the specific goals for our detection system are listed and the details regarding the object detection system that has been used for our experiments will be explained. Lastly, the dataset used for all training and testing will be shown.

Chapter 3

Neural Network for Polyp Detection

As explained in section 1.2, this thesis focuses on researching a potential improvement to the detection and automatic analysis subsystem of EIR. The current subsystem in EIR is split into two parts: *detection* and *localization*, but by adopting a neural network to perform detection, this system can detect irregularities in the video or image as well as finding the exact location the irregularities occur.

The main purpose of the detection system is to analyze video or images from examinations of the GI tract, in real-time. This work builds on the previous work of EIR, which is described in section 2.4.3, and hopes to further the research into a fast and efficient system for the detection and automatic analysis subsystem. To achieve this, the goals for the system are as follows:

- High disease detection accuracy.
- Real-time processing for supporting medical experts during colonoscopies.
- Efficient processing to allow for screening with VCE.
- Being expandable to allow for detection of different diseases.

The work conducted in this thesis consists mainly of running experiments on two detection systems, tiny-YOLO and YOLOv2. This chapter will give an overview of these two CNN object detection models and how they were setup for detecting polyps, followed by a complete overview of the base dataset used for training.

3.1 CNNs for Object Detection

For the experiments in this thesis, the chosen neural network architecture is YOLOv2 as well as a smaller (and faster) version called tiny-YOLO. All the experiments are performed using these two networks, so that their performance can be compared. Using YOLOv2 and tiny-YOLO was chosen for its reported high accuracy while still being able to perform detection in real-time. The YOLO architecture is implemented on custom neural network software: Darknet.

Figure 3.1: The Darknet Logo[1].

Darknet[2] is an open source neural network framework, that has been written in C and CUDA. The framework supports both CPU and GPU computation, but the use of CUDA means it cannot run directly on non-NVIDIA hardware. More about the hardware and frameworks that have been used can be found in section 4.1.1. Darknet was installed with two dependencies, OpenCV and CUDA, and was quite simple to install:

- First the github repository was cloned with the command: git clone `https://github.com/pjreddie/darknet.git`.
- Then Darknet had to be built by typing *make*.
- For this thesis, we built Darknet after enabling GPU and OpenCV in the Makefile[3].

Once Darknet had been built, with the said dependencies[4], there were changes that needed to be done before the network would be ready

[1] Logo is from the Darknet website: `https://pjreddie.com/darknet/`.

[2] Darknet website: `https://pjreddie.com/darknet/`.

[3] Before a GPU was obtained for the experiments, it was attempted to train the network by using the CPU. This folly was ended when it seemed that the training of the small network, tiny-YOLO, would last up to one week.

[4] As said: CUDA and OpenCV. The installation process is not particularly pleasant for either.

to begin training for our use case, meaning detecting of polyps. The changes performed are listed and explained in section 3.2.

3.1.1 Training Time and Pre-Trained Weights

The network is pre-trained on ImageNet [8] and the resulting weights are distributed along with the YOLO source code. Using the pre-trained weights, instead of random initialization when training, means that the network does not have to learn basic image features from scratch (see section 2.2.2). Using the pre-trained weights, reduces the time to convergence during training, meaning that they significantly lower the time required for training or fine-tuning the network towards detecting new objects.

3.1.2 Training Variation

Training CNNs with stochastic gradient descent is an optimization problem and there are no guarantees to find the global optimum, or to find the same results each time. This, coupled with the built-in data augmentation in YOLO, will give a slightly different model, every time it is trained.

The built-in data augmentation that YOLO performs during training includes:

Scaling and translations: Random scaling and translations of up to 20% of the original image size.

Rotation and cropping: Random rotation and performs random cropping of up to 80% of the image.

Color shifts: Random adjustment of the exposure and saturation of the image, up to a factor of 1.5 in the HSV [5] color space.

Due to the stochastic training process, there might be slight variation in the resulting models from the experiments performed for this thesis. To account for this the training data has only been changed slightly for each new iteration of experiments, in an attempt to reduce the differences in the achieved results and ensure that they are not just caused by random variances.

[5] Hue, saturation and value: https://en.wikipedia.org/wiki/HSL_and_HSV.

3.2 Configuring YOLO for custom objects

The reference implementation of the YOLOv2 network had a few hard-coded values that had to be changed to allow for starting the training for detection of custom objects, which in this case is *polyps*:

Editing the configuration file: The configuration file is the network description file i.e., it describes the network structure (meaning the convolutional and maxpooling layers) and contains technical specifications that Darknet reads. The technical specifications are values concerning the training scheme, which includes values like the learning rate (this is the rate of how much the weights are updated per iteration) and the total amount of training steps. When cloning the Darknet repository, several configuration files are included as well as the one used for our experiments: *yolo-voc.cfg* for YOLOv2 and *tiny-yolo-voc.cfg* for tiny-YOLO. There were two modifications necessary in this file, before the experiments could begin:

1. Within the configuration file we changed the filter value of the last convolutional layer, according to formula described in section 3.2.1.
2. The number of classes also had to be changed for our polyp detection use case, meaning we set the value of *classes = 1*.

Object Names: To allow the network to print out a bounding box with the corresponding object name, the file ending with *.names* has to contain the specific object name. This file only consists of one name: *polyp*.

Data File: The data file had to be changed to fit our setup. The syntax, and its specific content for our first experiment is explained in section 3.2.2.

Other than the hardcoded values that we had to change to fit our polyp detection use case, Darknet has its own unique label format. In section 3.3.1 this format is explained.

3.2.1 Outputs from YOLO

The figure 3.2 shows the new classification model at the base of YOLOv2, called Darknet-19. When it comes to training for detection this model has been modified by replacing the last convolutional layer with three 3×3 convolutional layers with 1024 filters followed by a final 1×1 convolutional layer with the number of outputs needed for detection. The formula for calculating filters corresponding to outputs:

$$(Classes + Coordinates + 1) \times NumberOfBoxes$$

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Figure 3.2: The Darknet-19 classification model[6].

For this thesis, where the number of classes were limited to our polyp detection use case, we only require filters corresponding to one class in the output layer. The network predicts 5 boxes with 4 coordinates each and 1 class per box, which gives $(1 + 4 + 1) \times 5 = 30$ filters for the output layer.

3.2.2 Data File

obj.data

```
1  classes= 1
2  train  = /home/torbjonh/Documents/YOLO/darknet/model1/YOLO/train_1.txt
3  valid  = /home/torbjonh/Documents/YOLO/darknet/model1/YOLO/test_1.txt
4  names = model1/YOLO/data/obj.names
5  backup = model1/YOLO/backup
6
```

Figure 3.3: The datafile from the first experiment.

The figure 3.3 shows the file that was used in the first experiment. In the *.data* file we specified the number of classes that was to be detected, the full system path to the training and validation text files, their format as explained in section 3.3.2, as well as the project paths to the name file and the backup directory. The backup directory is the file used to store the checkpoint weights that are created by the network during training.

[6]Figure from YOLOv2 paper [42].

3.3 The Dataset

For this thesis, the ASU Mayo Clinic polyp dataset[7] [56], is used as the training and testing dataset. This is the only publicly available dataset that contains a large amount of annotated data. The dataset contains 20 short colonoscopy videos, of which 10 contain polyps. Each frame in these videos comes with a corresponding ground truth image (or binary mask, see figure 3.4) that indicates the polyp region, if one exists. The resolution varies between 720×480 to 1920×1080 pixels. As seen from table 3.1, the videos are named using *np* (no polyps) or *wp* (with polyps).

This dataset was used as training and test data for all experiments that relates to the polyp detection use case in this thesis.

Video	Frames with Polyps	Frames Without Polyps	Total Number of Frames
ShortVD_np_5	0	682	682
ShortVD_np_6	0	838	838
ShortVD_np_7	0	769	769
ShortVD_np_8	0	712	712
ShortVD_np_9	0	1843	1843
ShortVD_np_10	0	1925	1925
ShortVD_np_11	0	1550	1550
ShortVD_np_12	0	1740	1740
ShortVD_np_13	0	1802	1802
ShortVD_np_14	0	1639	1639
ShortVD_wp_2	245	79	324
ShortVD_wp_4	910	0	910
ShortVD_wp_24	374	145	519
ShortVD_wp_49	391	110	501
ShortVD_wp_52	684	422	1106
ShortVD_wp_61	209	130	339
ShortVD_wp_66	234	184	418
ShortVD_wp_68	189	70	259
ShortVD_wp_69	235	381	616
ShortVD_wp_70	385	25	410
Total	3856	15046	18902

Table 3.1: The starting dataset.

3.3.1 Darknet Label Format

Darknet, sadly, cannot operate with the dataset as it is. In the format that is specific for Darknet, each image needs a corresponding ground truth label. So, instead of the binary ground truth images seen in figure 3.4, Darknet requires a *.txt* file for each image. This label file should contain a line for each ground truth object in the image:

$$ObjectClass \;\; X \;\; Y \;\; Width \;\; Height$$

[7]The ASU Mayo Clinic polyp dataset website: https://polyp.grand-challenge.org/site/Polyp/AsuMayo/.

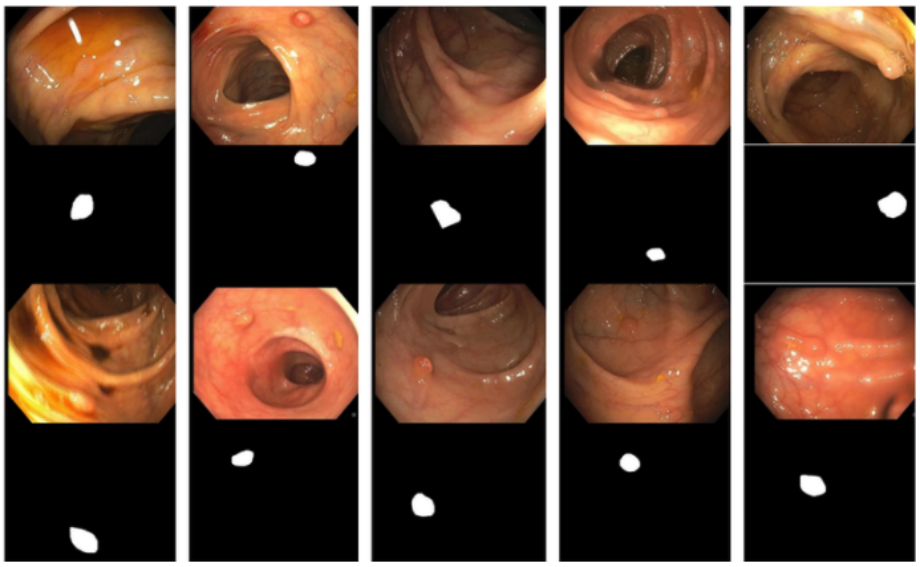


Figure 3.4: Frames from the ASU Mayo Clinic Polyp dataset and their corresponding ground truth images [56].

Where X, Y, Width and Height are relative to the width and height of the image. X and Y should also be the coordinates of the center point of the resulting bounding box [8].

3.3.2 Train & Valid Text Files

Together with the label files from section 3.3.1, Darknet requires one separate text file for the validation and training dataset, as seen in figure 3.3. These text files contain the full system paths to the training and validation images, one image path per line. Further, the *images/* and *labels/* directories should be placed within the same parent directory, as Darknet simply replaces "images" with "labels" when looking for the training and validation text files.

3.3.3 Data Converting

The polyp dataset is a set of 20 videos and contains a binary ground truth mask for each frame of these videos. To modify the data to work with the Darknet label format it was necessary to save each single video frame from the colonoscopy videos, which was done using a simple python script with OpenCV. Another python script was used to generate the label files, format as specified in section 3.3.1, from the

[8] A bounding box is the smallest possible rectangle that encompasses the object.

corresponding binary ground truth images[9].

3.4 Summary

This chapter lists the goals for the system and gives an overview of the object detection system, which includes a how-to on configuring YOLO for running detection with custom objects. The dataset that have been used for all training and testing, the ASU Mayo clinic polyp dataset, is also presented.

In the following chapter, there will be a detailed presentation of how the experiments were conducted, before the experiments are presented and the results are discussed.

[9] All code used in this thesis can be found in section 4.1, which provides a link to GitHub.

Chapter 4

Experiments and Discussion

This chapter provides useful context for how the experiments have been conducted, through first presenting the hardware and frameworks that have been installed as well as specifics on how to run the training and validation of experiments. The chapter then moves on to listing the exact training and validation datasets used. Further, the results that have been achieved from the experiments are listed and continues with presenting the experiments conducted in this thesis, as well as a discussion about the achieved results. Lastly, there is a general discussion about the experiments, observations from these experiments and the goals for the detection system. The chapter ends with a section that lists a few lessons that have been learned during the course of this thesis.

4.1 Setup

All code, data and result files are available on GitHub[1]: `https://goo.gl/zgEWJU`.

4.1.1 Hardware & Frameworks

All experiments have been executed with the same version of Darknet, using commit **b61bcf5**, by cloning the Darknet GitHub repository, as explained in section 3.1. Following these steps, Darknet was compiled with CUDA 8.0 and OpenCV 2.4.13 enabled. The hardware that was used for the experimenting was a single desktop computer with an Nvidia GeForce GTX 1070 GPU and running Kubuntu 16.04.

[1]Full link to the GitHub page here: https://github.com/Tobzor/PillCamMaster.

4.1.2 Running Experiments

After the preparation of the dataset and the modifications to Darknet, which can be found in sections 3.3 and 3.2, respectively, all that was required was starting the training of the detection system. To start training, the following command was executed:

```
./darknet detector train c.cfg d.data w.weights
        > training-output.log
```

Where the *.cfg* and *.data* files are specified in section 3.2. The *.weights* file can be one of the provided pre-trained models, for example *darknet19_448.conv.23*, or one of the checkpoint weights saved in the backup directory while training.

The *train* command will start the network training process and store the training output into the output file (here: *training-output.log*) and after a set amount of iterations, meaning the training steps set in the configuration file from section 3.2, the network saves checkpoints of the weights in the backup directory at the location that has been set in the Data file from section 3.2.2. Any of these checkpoint weights from the backup directory can then be used for detection (or as a checkpoint from where to resume the training, hence the name).

To evaluate the performance of the trained network, with its weights, we executed the following command:

```
./darknet detector valid c.cfg d.data final.weights
```

This validation command uses the *valid* path value that has been set in the data file, from section 3.2.2. The valid command outputs one result text file per class. This text file then contains all the bounding boxes that the network predicted, in the format specified by PASCAL Visual Object Classes (PASCAL VOC) [10]:

$$ImageID\ \ Probability/Confidence\ \ Xmin\ \ Ymin\ \ Xmax\ \ Ymax$$

4.2 Dataset reference

An overview of the training and testing dataset that have been used in our experiments can be found in tables 4.1 and 4.2, respectively. These tables list the dataset videos as well as useful statistics concerning the number of positive, negative and the total number of video frames each video contains.

Starting Dataset	Frames with Polyps	Frames Without Polyps	Total Number of Frames
ShortVD_wp_49	391	110	501
ShortVD_wp_52	684	422	1106
ShortVD_wp_61	209	130	339
ShortVD_wp_66	234	184	418
ShortVD_wp_68	189	70	259
ShortVD_wp_69	235	381	616
ShortVD_wp_70	385	25	410
Added in the second experiment.			
ShortVD_np_5	0	682	682
ShortVD_np_6	0	838	838
ShortVD_np_7	0	769	769
ShortVD_np_8	0	712	712
Added in the third experiment.			
flips70-49	4654	2644	7298
Total in **Starting Dataset**	2327	1322	3649
Total in **Second Experiment**	2327	4323	6650
Total in **Third Experiment**	6981	6967	13948

Table 4.1: The training dataset, with each new addition of data, for all experiments.

Video	Frames with Polyps	Frames Without Polyps	Total Number of Frames
ShortVD_np_9	0	1843	1843
ShortVD_np_10	0	1925	1925
ShortVD_np_11	0	1550	1550
ShortVD_np_12	0	1740	1740
ShortVD_np_13	0	1802	1802
ShortVD_np_14	0	1639	1639
ShortVD_wp_2	245	79	324
ShortVD_wp_4	910	0	910
ShortVD_wp_24	374	145	519
Total	1529	10723	12252

Table 4.2: The testing dataset for all experiments.

4.3 Results

This section presents the overall detection performance that the tiny-YOLO and YOLOv2 networks achieved, for each experiment.

Experiment	TP	FP	TN	FN	Recall	Precision	F1 score
First - Base Dataset 4.6	1130	413	10397	399	73.90%	73.23%	73.57%
Second - Negative Training 4.7	934	318	10596	595	61.09%	74.60%	67.17%
Third - Data Augmentation 4.8	1126	396	10598	403	73.64%	73.98%	73.81%

Table 4.3: Overall results from YOLOv2 using the FINAL.weights.

Table 4.3 lists the results that have been achieved, for each experiment, by YOLOv2. Table 4.4 presents the same, but for tiny-YOLO. The tables list the evaluation metrics that were received when running the evaluation, which is explained in section 4.4. Discussions and more specific metrics are presented for the experiments in their own sections, which is referred to in the table.

From the performance evaluation section 2.3, it is possible to understand the performance metrics that have been presented in tables 4.3

Experiment	TP	FP	TN	FN	Recall	Precision	F1 score
First - Base Dataset 4.6	971	869	10121	558	63.51%	52.77%	57.64%
Second - Negative Training 4.7	903	441	10505	626	59.06%	67.19%	62.86%
Third - Data Augmentation 4.8	1146	910	10422	383	74.95%	55.74%	63.93%

Table 4.4: Overall results from tiny-YOLO using FINAL.weights.

and 4.4. The true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN) from the tables are used to calculate the recall and precision that the system achieves. Recall and precision are then again used to calculate the F1-score, following the equations in section 2.3. When comparing the systems detection performance, we use the recall and precision to calculate the F1-score as this is the easiest metric to compare with. F1-score is basically the probability that the predictions, that are made by the network, is correct. Table 4.5 shows the tiny-YOLO and YOLOv2 networks and their different performances with a change in network resolutions. These differences are discussed further in section 4.8.

System	Recall	Precision	F1-score	FPS
Tiny-YOLO	74.95%	55.74%	63.93%	123
Tiny-YOLO-608	81.82%	49.54%	61.72%	75
Tiny-YOLO-832	79.79%	46.41%	58.68%	51
YOLOv2	73.64%	73.98%	73.81%	51
YOLOv2-608	80.18%	67.40%	73.24%	29
YOLOv2-832	77.83%	62.27%	69.19%	19

Table 4.5: The table shows tiny-YOLO and YOLOv2 with the change in detection performance over different network resolutions.

4.4 System Evaluation Script

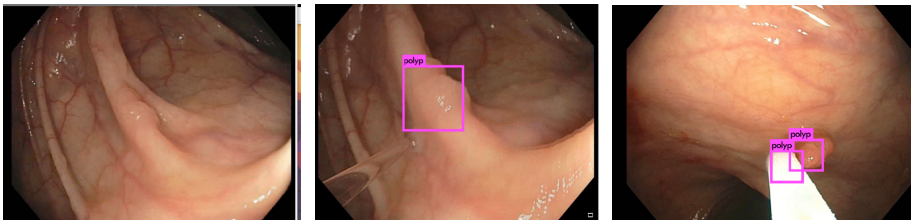
The performance evaluation was done by comparing the resulting file from running the *valid* command (from section 4.1.2) in Darknet, with the ground truth labels from our validation dataset. This is done by the *eval_testing.py* script created for this purpose[2]. Through running our script several times and observing the results, these are the values that it was decided to use:

Probability Threshold: This threshold is set to 10%, to filter out the more uncertain predictions. The *valid* command outputs a resulting file that contains the image, prediction probability and bounding box coordinates, as seen in section 4.1.2. It is this prediction probability that the threshold is for.

[2]Link to GitHub can be found in section 4.1.

IOU Threshold: As explained in section 2.3, a typical threshold for the IOU is 50%. In our evaluation, this has been reduced to 40% because of the small size of the bounding boxes that are used.

Duplicate Detections: Multiple detections, on the same image, will result in one true positive (if it exists) and the rest will be set as false positives. What has not been taken into consideration is the way YOLO runs its detections. By separating the image into region boxes, a polyp (it's center) could be located on the border between these regions which could, potentially, create two probable detections within that image. See figure 4.1, for some example predictions made by the smaller network, tiny-YOLO.



(a) False negative. (b) True positive. (c) Duplicate detection.

Figure 4.1: A sample of a false negative, true positive and duplicate predictions by tiny-YOLO.

4.5 Experiments

There are several experiments that have been conducted in this work, all related to the GI tract use case, concerning an automatic disease detection system. By using the architecture of the two networks, tiny-YOLO and YOLOv2, there are many parameters that can be varied and tested, but at first the focus will be exploring different strategies for the training dataset. As mentioned in section 1.4, the experiments will follow an iterative approach.

When training the networks for our experiments, there were some differences in the training scheme between tiny-YOLO and YOLOv2. YOLOv2 had a learning rate of 0.0001 and 45000 training steps, whereas tiny-YOLO had a learning rate of 0.001 and 40100 training steps. Every evaluation has been run using the *final.weights* from each model, which are created after the last training step, as these were the weights that gave the best results on the testing dataset.

The main experiments conducted in this thesis will be presented, and their results will be discussed, in the following sections 4.6, 4.7, 4.8

4.6 First - Base dataset

The first experiment was conducted with the starting dataset, after the conversion to the Darknet format, as it is explained in section 3.3.

The dataset from table 3.1 was split into two parts, training and test data. The training data was constructed from the images and labels from ShortVD_wp_49 down to ShortVD_wp_70 and the testing dataset was constructed from video ShortVD_np_9 down to ShortVD_wp_24, leaving the videos ShortVD_np_5 through ShortVD_np_8 unused. This splitting of the dataset was meant to leave as much of the data as possible for the training dataset, while still leaving enough positive data for the testing dataset.

The purpose with this experiment was to get a feel of how the two networks, YOLOv2 and tiny-YOLO, could perform when applied to detection *inside* the GI tract, as opposed to detecting objects on the streets (e.g. cars, planes, etc.), and how they were in use.

Discussion - Base Dataset

The main purpose of this experiment, as stated in section 4.6, was to determine how the two YOLO networks could perform when adopted for the polyp detection use case, by training the networks on the starting dataset, specified in table 4.1.

The achieved performance for YOLOv2 was 73.90% recall, 73.23% precision and a F1-score of 73.57%. Compared to the 63.51% recall, 52.77% precision and F1-score of 57.64% from tiny-YOLO, it shows that YOLOv2 achieves an overall better performance using the same data.

Looking at the metrics in table 4.6, it is quite clear that tiny-YOLO has overall lower performance than YOLOv2, particularly the precision the network reaches. Nevertheless, when examining the results, it is evident that the YOLO-networks have good potential for the polyp detection use case.

As one of the main goals for the disease detection is to achieve good detection accuracy, the next experiments was an attempt to first reduce the amount of false predictions, meaning false positives and false negatives, before increasing the amount of correctly detected polyps.

Metrics from evaluation	Tiny-YOLO	YOLOv2
Total number of predictions	31913	9149
Predictions below probability threshold	30073	7606
Classified as positive events	1840	1543
Total number of polyps in dataset	1529	1529
True positives	971	1130
False positives	869	413
Classified as negative events	10679	10796
True negatives	10121	10397
False negatives	558	399
Recall	63.51%	73.90%
Precision	52.77%	73.23%
F1-score	57.64%	73.57%

Table 4.6: Performance metrics from running evaluation of the first experiment.

4.7 Second - Negative training

For this experiment the unused negative (no polyp) videos from the dataset are added to the training data. From the Darknet forum, which is a Google Group created for the users of Darknet, one of the authors of YOLO, Joseph Redmon [42] (known as "SuperDragon McFuzzypants" in the forum), explains that adding negative images, with empty labels, can help the network to filter out unwanted detections. Specifically, he states that[3]:

> *"[...]. The only real reason to include them (meaning negative data) is if there is some object you don't want to detect that looks similar to objects you DO want to detect."*

And as one can see from the figure 3.4, it is not always easy to find a clear indication that there is a polyp in the images, which fits well with the mentioned quote. So, the purpose for this experiment was to reduce the amount of false predictions from the detection system. The unused videos negative videos from the dataset in table 3.1, videos ShortVD_np_5 through ShortVD_np_8, was added to the training dataset described in section 4.6.

Discussion - Negative Training

The purpose of this second experiment, as explained in section 4.7, was to focus on reducing the amount of false predictions by adding the

[3]Direct link to the Q&A in the Darknet forum: https://groups.google.com/d/msg/darknet/Bs4N3IiBH3s/EivXBkwzAwAJ.

unused *np* videos to the training data (see table 3.1) to help the networks become more familiar with the features in the GI tract. Potentially, this could help with filtering out unwanted predictions and make the networks better able to discern *polyps* from the GI wall[4].

At first glance it seems that the networks have reduced the total number of predictions when running detection, as can be seen in table 4.7. In the case of tiny-YOLO, the total amount of predictions has been reduced by almost 50%. This does however seem to have had some negative consequences for the recall achieved by the detection systems. As said in section 2.3, recall is based on the number of polyps detected out of the total number of possible polyps to detect.

Metrics from evaluation	**Tiny-YOLO**	**YOLOv2**
Total number of predictions	15408	7646
Predictions below probability threshold	14064	6394
Classified as positive events	1344	1252
Total number of polyps in dataset	1529	1529
True positives	903	934
False positives	441	318
Classified as negative events	11131	11191
True negatives	10505	10596
False negatives	626	595
Recall	59.06%	61.09%
Precision	67.19%	74.60%
F1-score	62.86%	67.17%

Table 4.7: Performance metrics from running evaluation of the second experiment.

Looking at the results in table 4.7, there is strong indication that introducing negative data into the training dataset had an undesirable outcome. A possible explanation might be that the total dataset that has been used in this experiment, which can be seen in table 3.1, is imbalanced, meaning the negative training data compared to the amount of images containing polyps is in a ratio of approximately 1:0.5 (negative:positive).

Investigating the ratio of positive and negative training data and what effects a variation of this ratio might have, the next experiment introduces data augmentation, which is performed on the positive training data. The purpose of this was to add more variation into the positive training data, which hopefully would let the networks generalize more to the different appearances, locations and angles that the diseases, in this case polyps, might be exhibiting in the GI tract.

[4]Or the mucosa, which is the innermost layer of the gastrointestinal tract.

4.8 Third - Data Augmentation

Because deep neural networks require a large amount of training data to achieve good detection performance and, unfortunately, the dataset available for this thesis is limited. To help remedy this, data augmentation was performed on the positive training data, following the work of Krizhevsky et al. [25]. This was to both improve upon the variability of the data as well as, potentially, boosting the detection performance the detection networks can achieve.

As said in section 3.1.2, YOLO already has several built-in data augmentation methods. Nevertheless, increasing the total size of the training dataset helps provide the network with more variances and helps reduce overfitting the model to the training data. Specifically, the data augmentation implemented was horizontal and vertical flips of the positive training data, meaning videos ShortVD_wp_49 to ShortVD_wp_70 from table 3.1.

Discussion - Data Augmentation

As said previously in section 4.8, the purpose of this experiment was to increase the amount of positive data in the training dataset. The experiment is based on the hypothesis that having a 1:1 ratio of positive and negative data with more variations of the positive data will increase the recall that the networks achieve.

Having a more balanced dataset, and more variations of positive data, has increased the detection performance of both networks. In the case of tiny-YOLO, the network actually surpassed its larger counterpart YOLOv2. This might have several explanations. One possible explanation for this might be drawn from the architecture differences in tiny-YOLO versus YOLOv2, meaning that tiny-YOLO have less convolutional layers and therefore have generalized better towards detecting previously unseen polyps.

Another potential explanation could be drawn from the training data. There are a total of 7 unique polyps within the training data and 3 unique polyps in the testing data. Considering that YOLOv2 is *deeper* than tiny-YOLO, YOLOv2 learns richer, more complex, features beyond the scope of tiny-YOLO, which might mean YOLOv2 has started to overfit to the training data and this increases the generalization error on new data.

However, YOLOv2 and tiny-YOLO also has a difference in learning rate (see section 4.5, which might mean that YOLOv2 simply was not done training after the predefined steps (although for one class this should be

more than enough, according to users from the Darknet forum[5]).

Metrics from evaluation	Tiny-YOLO	YOLOv2
Total number of predictions	15078	8214
Predictions below probability threshold	13022	6692
Classified as positive events	2056	1522
Total number of polyps in dataset	1529	1529
True positives	1146	1126
False positives	910	396
Classified as negative events	10805	11001
True negatives	10422	10598
False negatives	383	403
Recall	74.95%	73.64%
Precision	55.74%	73.98%
F1-score	63.93%	73.81%

Table 4.8: Performance metrics from running evaluation of the third experiment.

Comparing the results from the second experiment (table 4.7) and the results in table 4.8, tells us that the balancing of the dataset had the intended positive effect on the recall and precision metrics.

As pointed out in the general discussion section 4.10.1, random rescaling of the network during training was disabled for YOLOv2 during the first experiments, and was not noticed until after the results from this experiment was compared. What was interesting, was that after the retraining with random rescaling (see multi-scale training in section 2.5.4) there were no significant changes in performance. However, this allows for a manual change in network-resolution that can be set in the configuration file and the standard value here is 416×416. Increasing the resolution makes the network more precise and allows for better detection of small objects, up to a certain resolution before the precision drops[6]. As expected, the increased resolution also makes detection significantly slower in terms of FPS. Results from running detection on different network resolutions are shown in table 4.9.

However, when the resolution is increased from the standard 416×416, the networks has a decrease in precision. A probable explanation for this can be drawn from the total number of predictions, where there is an increase in number of predictions for every increase in resolution. Something that would be interesting to test further, is how the networks react to different input image resolutions when running detection on

[5]Darknet forum: https://groups.google.com/forum/#!forum/darknet.

[6]The YOLO training scheme only uses random rescaling between resolutions of 320-608.

Metrics from multi-scale detection	Tiny-YOLO		YOLOv2	
Resolutions	608 × 608	832 × 832	608 × 608	832 × 832
Frames Per Second	75	51	29	19
Total number of predictions	29462	41657	9460	10626
Predictions below probability threshold	26937	39028	7641	8715
Classified as positive events	2525	2629	1819	1911
Total number of polyps in dataset	1529	1529	1529	1529
True positives	1251	1220	1226	1190
False positives	1274	1409	593	721
Classified as negative events	10540	10540	10832	10812
True negatives	10262	10231	10529	10473
False negatives	278	309	303	339
Recall	81.82%	79.79%	80.18%	77.83%
Precision	49.54%	46.41%	67.40%	62.27%
F1-score	61.72%	58.68%	73.24%	69.19%

Table 4.9: Performance metrics from running evaluation of the third experiment with different resolutions for both tiny-YOLO and YOLOv2.

different network resolutions.

4.9 Comparing with EIR

System	Recall	Precision	F1-score	FPS
EIR	88.90%	96.40%	91.60%	~ 75 [a]
Deep-EIR	87.9%	87.2%	87.6%	30
Tiny-YOLO	74.95%	55.74%	63.93%	123
Tiny-YOLO-608	81.82%	49.54%	61.72%	75
Tiny-YOLO-832	79.79%	46.41%	58.68%	51
YOLOv2	73.64%	73.98%	73.81%	51
YOLOv2-608	80.18%	67.40%	73.24%	29
YOLOv2-832	77.83%	62.27%	69.19%	19

Table 4.10: EIR versus the YOLO detection systems.

[a] 300 FPS for detection and 100 FPS for localization.

Comparing our results with that of EIR, which is presented in section 2.4.3, it would seem that their detection performance is superior to the performance tiny-YOLO and YOLOv2 reaches. The detection comparing was difficult considering that the normal EIR uses handpicked features in a machine learning approach and Deep-EIR uses a different dataset. Nevertheless, the results in table 4.10 gives an approximate overview of how the evaluated systems performs compared to those from EIR.

Deep-EIR (results shown in table 4.10) is a proof-of-concept of EIR for multi-disease classification based on deep learning. In particular, it is

based on the Inception v3 architecture [54]. Deep-EIR is presented in the PhD thesis of Michael Riegler [46][7] and the evaluation of its accuracy have been completed using a multi-class dataset that contains a total of 300 images, with 50 images for the 6 different classes.

Tiny-YOLO and YOLOv2 detects approximately the same number of polyps, seen from the recall metrics, but with a varying degree of precision. As explained in section 2.4.3, precision is a value that shows how many of the predictions a system performs are correct. As seen from the F1-score in table 4.10, the highest score that has been reached with our detection systems, 73.81% still has potential for improvement when seen in the light of both EIR systems, 91.60% for EIR and 87.6% for Deep-EIR.

Unfortunately, there was not enough time at the end of this thesis to run a leave-one-out cross-validation to get an exact comparison between the YOLO networks and the handpicked features of EIR, nor was there time to run a multi-class experiment using the multi-class dataset that was used in Deep-EIR.

4.10 General Discussion

The idea for this thesis was to explore the viability of using modern object detection systems as a medical assist and automatic analysis tool, following the work of EIR, which is presented in section 2.4.3. The achieved results suggest that using a CNN object detection system for automatic disease detection is viable if there exists access to good training data. However, the approach of using tiny-YOLO and YOLOv2 requires further investigation into how the networks generalize to new data as well as some form of cross-validation to gain a clearer view of how the networks could perform.

Preliminary results achieved from the experiments in this thesis suggests that there is potential for tiny-YOLO or YOLOv2 as a detection subsystem within the EIR system (presented in section 2.4.3). This section presents some interesting observations noticed during the experiments and discusses if the goals for the system, presented in section 1.2 and in the introduction of chapter 3, have been reached.

4.10.1 Training Observations

As the results from tiny-YOLO was compared to YOLOv2, there were some interesting behavior that was observed between the two networks.

[7] Section 3.6.2 in the paper.

To pinpoint this difference in behavior the configuration files for the networks were compared.

When examining the YOLOv2 configuration file that was used for the experiments, it was noticed that the random rescaling of the network during training was *disabled*. As mentioned in section 2.5.4, multi-scale training makes the network robust for images in different sizes. Considering that the training dataset contains images in many different sizes and the fact that tiny-YOLO, which has this enabled, achieves higher recall than YOLOv2 without random rescaling, the random rescaling for YOLOv2 was enabled and the network was retrained for the third experiment in section 4.8.

The multi-scale training did not provide any significant difference at first glance. Instead the random rescaling of the network during training allows for a network that is more robust towards detection across different input resolutions. This, in turn, made it possible to change the network resolution[8] when running detection. The results from running detection at different resolutions, are presented in table 4.9.

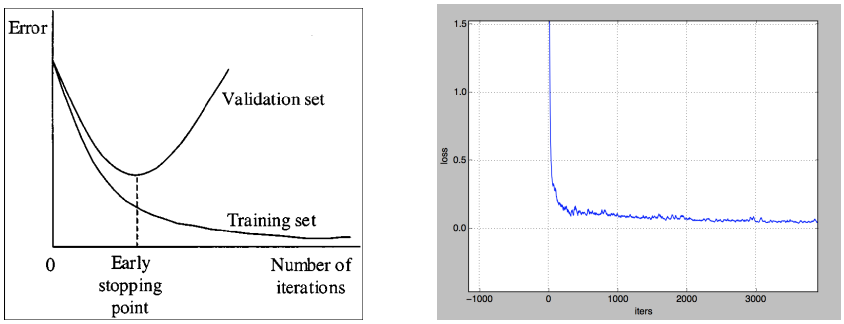
As is mentioned in section 2.5.5, using tiny-YOLO as well as YOLOv2 could give valuable insight into how the networks operate and make it possible to investigate if tiny-YOLO could be used as a proxy for YOLOv2, to allow for faster iteration conducting experiments. As we have observed by running the experiments presented in sections 4.6, 4.7 and 4.8, the networks have an approximate matching of increase and decrease in their detection performance throughout the different experiments. This observation suggests that tiny-YOLO, with its approximate 1/3 of the training time for YOLOv2, can be used as a proxy for running more iterations (and variations) of experiments. However, it should be noted that this is based on the experiments conducted in this thesis and it might be acting differently on other types of experiments, or even with other software versions, which suggests that there is some need for further investigation.

Everything that was read about training towards detecting objects with Darknet and its predefined network structures, YOLOv2 and tiny-YOLO, was pointing towards needing less training steps when training to detect a single class of objects. From the GitHub of AlexeyAB[9] he states that it is usually sufficient with 2000 training steps, as opposed to the 40100 steps for tiny-YOLO and 45000 for YOLOv2. For a more precise definition when to stop training, it is possible to use the training output from Darknet. Specifically concerning the average loss (error)

[8] By network resolution we mean that that is the resolution of which the input images are scaled down (or up) to when performing detection on said images.

[9] AlexeyAB at GitHub: https://github.com/AlexeyAB/darknet#when-should-i-stop-training.

printout and "stopping" training (using the checkpoint weights) at the early stopping point. See figure 4.2a.



(a) (b)

Figure 4.2: (a) The figure to the left depicts the early stopping point[10]. (b) The figure on the right depicts the loss plot from tiny-YOLO.

However, when the loss was plotted for one of our training runs with tiny-YOLO, seen in figure 4.2b, it was rather difficult to draw any useful information from the graph considering we had no validation set to see when the network started to overfit on the training data. Instead, we tried several of the checkpoint weights that are saved during training. Unfortunately, this was proven to be unnecessary as the results we achieved, meaning the F1-score, which takes into account both recall and precision, was higher when measured from the final weights created after the completion of the training steps.

A potential explanation for not getting a higher F1-score when using any of the checkpoint weights, could be related to our dataset. It is possible that overfitting on the training data helps the network towards better recognizing the polyps in the testing data, because both training and testing datasets are just different parts out of the same dataset. This would, however, be interesting to investigate further by measuring the generalization loss on a validation dataset and plotting this together with the training loss to see if or when overfitting occurs.

4.10.2 System Goals Achieved?

The overall detection performance that has been achieved is presented in tables 4.3 and 4.4. The evaluation of the detection performance has been done by running detection on the test dataset, shown in table 4.2, and then evaluating using the evaluation script presented in section 4.4.

The goal of reaching high disease detection accuracy could be said to have been reached. Although, the tested systems, tiny-YOLO

[10]Early stopping is a form of regularization used to avoid overfitting.

and YOLOv2, still have potential for improvement compared to other approaches from EIR[11]. As seen in table 4.10, the detection accuracy achieved from the experiments conducted in this thesis is lower than that of EIR and Deep-EIR. The relevant results from EIR is presented and compared to our detection systems in section 4.9.

Further, the goals of running the detection system in real-time and being efficient at handling large amounts of data (for VCE screening) have been reached. The large network, YOLOv2, reported a total detection time of 239 seconds on the validation dataset, which contains 12252 images, and this gives a rate of 51.26 FPS. Which is above the defined real-time border of 25-30 FPS. The small network, tiny-YOLO, reported its total detection time to being 99 seconds on the validation dataset. This gives a rate of 123.75 FPS, which is 4$\times$ faster than the real-time border. Note that the FPS was reduced when running detection on larger resolutions, which is shown in table 4.9.

The goal of the system being easily expandable have, unfortunately, not been addressed due to lack of time. However, the YOLO papers concerning the detection system [41, 42] provides information about multi-class detection and as a matter of fact, the reference networks of YOLO is already set up for the detection of multiple classes and fully trained models (or weights) are provided via the Darknet website[12]. Unfortunately, this does not prove that YOLO is viable for detecting more diseases than polyps, which suggests that further investigation into multi-class detection is required, preferably using the same dataset from Deep-EIR[13] to allow a better comparison.

4.10.3 Lessons Learned

As this thesis was *my* first research experience with the conducting of larger experiments, the idea with this section is to note some of the things learned during the course of this thesis.

Conducting experiments when using deep learning requires a lot of time to complete, which was a new experience. Usually, when testing different iterations of some practical work it uses a few minutes to run through the process before its done. Adjusting to working with deep learning, which uses between 5 to 72 hours, required a lot of try and fail runs to determine a process that worked for conducting the experiments. However, what was found lacking in hindsight, when writing, was the documentation of the process and results during the

[11] EIR has been presented in section 2.4.3.

[12] https://pjreddie.com/darknet/yolo/.

[13] Deep-EIR is a proof-of-concept using deep learning with EIR and is presented in the EIR papers found in section 2.4.3.

experiments, which unfortunately meant that some of the results had to be rechecked when the writing started.

Another lesson learned over the course of this thesis was that figuring out what system to use and how to conduct experiments, should be done after the general goals for the system have been set. Especially important, if the system is to be compared with previous work (as in this thesis) leave enough time during the experimental phase to run an approximately similar experiment, e.g. a leave-one-out cross-validation, as this would make running a comparison a lot easier.

4.11 Summary

As discussed in section 4.10.2, most of the goals listed in section 1.2 relating to our system was reached. The system still has room for improvement in the case of high disease detection accuracy and expanding the detection for more than one class of disease have not been addressed. Nevertheless, the proposed systems achieve good results on the dataset used, which suggests that further investigation into the use and possibilities of Darknet, specifically tiny-YOLO and YOLOv2, should be considered. We found that the networks tested was a little lacking in the hope of high detection accuracy, but the systems, particularly tiny-YOLO, shines when it comes to speed.

In the next chapter, we summarize the thesis and its research contributions and draw a conclusion, which is then followed up with suggestions for further work.

Chapter 5

Conclusion

In this chapter, we summarize the work that has gone into this thesis and what has been done to address the problem statement in section 1.2. The chapter then presents the main contributions of this thesis and what future work remains to be done.

5.1 Summary

In this thesis, we presented our experiences with researching a new potential technology to be used for automatic analysis of medical imaging of the GI tract. This technology, which is using deep learning for object detection, was aimed at being a potential improvement in EIR (presented in section 2.4.3) as the detection and automatic analysis subsystem. We concluded that the used object detection systems, meaning both tiny-YOLO and YOLOv2, can be used for automatic detection of diseases in the GI tract. However, while both systems reached a satisfactory detection rate and were very fast, particularly tiny-YOLO, when running detection, they did not reach the same level of detection rate and accuracy as was reported in EIR. This suggests that there is still potential for improvements when adopting the YOLOv2 and tiny-YOLO object detection systems for automatic disease detection in the GI tract.

At the beginning of this thesis, meaning in chapter 2, we introduced the medical scenario in section 2.1, which is relevant for examining the GI tract (can be seen in figure 2.1) and then proceeded to present the technology that has the potential to significantly improve upon this medical scenario, namely deep learning, which can be found in section 2.2. After introducing some of the relevant research for automatic polyp detection in section 2.4, the chapter continues with a presentation of several object detection systems in section 2.5, leading

up to the YOLOv2 system, in section 2.5.4, that was used to conduct experiments in this thesis.

In chapter 3, the introduction gives a presentation of the goals we have for the detection system and the chapter continues with explaining the detailed steps necessary to make polyp detection work within the chosen object detection system in section 3.1. Lastly, the section 3.3 presents the ASU Mayo Clinic polyp dataset [56] and explains the data format that Darknet, which the chosen detection system YOLOv2 is built on, requires.

Chapter 4 continues on from where the previous chapter ended and begins with the setup, in section 4.1, used for running the experiments in this thesis. The chapter then presents statistics for both splits of the dataset that has been used as training and testing data, in section 4.2. Chapter 4 then provides the overall results, in section 4.3, that has been achieved through the experiments conducted in this thesis as well as how the system was evaluated, which can be found in section 4.4.

At the end of chapter 4 the experiments is presented, and their results are discussed, in their own sections 4.6, 4.7 and 4.8. We then proceed to compare our systems performance with that of EIR in section 4.9, before moving on to a more general discussion in section 4.10. This section looks into our systems performance, observations that has been made during training and a discussion about the original goals we set for our system and if they were reached.

5.2 Main Contributions

In this thesis, we have shown that deep learning can be used for automatic detection of diseases in the GI tract. The preliminary results achieved, by using the YOLO object detection system, suggests that there is large potential for object detection systems within automatic analysis of medical imaging, specifically the detection of diseases in the GI tract. The systems both reach good detection accuracy, while still being within the real-time border of 30 FPS that was defined in section 1.2. Using either tiny-YOLO or YOLOv2 is possible, as tiny-YOLO runs detection fast (at 123 FPS), but has less accuracy, meaning that tiny-YOLO generates more false positives than the larger YOLOv2 network. This makes it possible to trade between accuracy and speed, by using either tiny-YOLO for its speed, or YOLOv2 for its precision (less false positives).

The main contribution in this thesis is the research and evaluation of a new, and potential improvement, to the detection and analysis subsystem in the multimedia system EIR. This system has proven able to detect polyps, with good accuracy, that can be seen in GI tract

examination videos, either from a VCE or during manual procedures. It is, however, important to point out that the used dataset is limited in its size and that evaluations, and further experiments, on a larger amount of data is recommended.

5.3 Future Work

For future work, there are two new datasets that have become available: an extended multi-class dataset for computer aided GI tract detection called Kvasir [36] and a bowel (colon) preparation quality video dataset called Nerthus [37]. Using these datasets to either test, train or validate the detection systems would be very interesting and could be used to investigate if the detection systems really have generalized its weights towards new, previously unseen data. Further, running cross-validation using the ASU Mayo Clinic polyp dataset to provide a more detailed comparison between other detection systems as well as using the multi-class dataset, that was mentioned in section 4.8, to see how the detection systems could perform on multiple types of diseases is highly recommended as future work.

The results achieved in this thesis are from running three iterations of experiments and investigating the results from manipulating or augmenting the training dataset. Even from a pure data augmentation angle, there are many more experiments that could improve upon the detection systems performance. For example, more pre-processing of the training data by applying different filters or a smoothing of the bright areas within the data that occur due to screen glare from the VCE light source, which appear because the camera and light source is so closely placed. Even an investigation into how the detection system acts without the built-in data augmentation could be relevant and provide context as to how the system would act in a real setting.

Bibliography

[1] David W. Aha, Dennis Kibler, and Marc K. Albert. "Instance-based learning algorithms". In: *Machine Learning* 6.1 (1991), pp. 37–66. DOI: 10.1007/BF00153759. URL: http://dx.doi.org/10.1007/BF00153759.

[2] Luís A Alexandre, Joao Casteleiro, and Nuno Nobreinst. "Polyp detection in endoscopic video using SVMs". In: *Proc. of PKDD*. 2007, pp. 358–365.

[3] Stefan Ameling, Stephan Wirth, Dietrich Paulus, Gerard Lacey, and Fernando Vilarino. "Texture based polyp detection in colonoscopy". In: *Bildverarbeitung für die Medizin*. Springer, 2009, pp. 346–350.

[4] Wendy S Atkin, Rob Edwards, Ines Kralj-Hans, Kate Wooldrage, Andrew R Hart, John MA Northover, D Max Parkin, Jane Wardle, Stephen W Duffy, and Jack Cuzick. "Once-only flexible sigmoidoscopy screening in prevention of colorectal cancer: a multicentre randomised controlled trial". In: *The Lancet* (May 2010). DOI: 10.1016/S0140-6736(10)60551-X. URL: http://dx.doi.org/10.1016/S0140-6736(10)60551-X.

[5] Da-Chuan Cheng, Wen-Chien Ting, Yung-Fu Chen, Qin Pu, and Xiaoyi Jiang. "Colorectal polyps detection using texture features and support vector machine". In: *Advances in Mass Data Analysis of Images and Signals in Medicine, Biotechnology, Chemistry and Food Industry*. Springer, 2008, pp. 62–72.

[6] Francesco Ciompi, Kaman Chung, Sarah J van Riel, Arnaud Arindra Adiyoso Setio, Paul K Gerke, Colin Jacobs, Ernst Th Scholten, Cornelia Schaefer-Prokop, Mathilde MW Wille, Alfonso Marchiano, et al. "Towards automatic pulmonary nodule management in lung cancer screening with deep learning". In: *arXiv preprint arXiv:1610.09157* (2016).

[7] D. E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, and Paul R. Young. "Computing As a Discipline". In: *Communnications of the ACM* 32.1 (Jan. 1989). Ed. by Peter J. Denning, pp. 9–23. ISSN: 0001-0782. DOI: 10.1145/63238.63239. URL: http://doi.acm.org/10.1145/63238.63239.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *Proc. of IEEE CVPR*. 2009, pp. 248–255.

[9] American Society For Gastrointestinal Endoscopy. *Understanding Capsule Endoscopy*. URL: https://www.asge.org/home/for-patients/patient-information/understanding-capsule-endoscopy.

[10] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. "The Pascal Visual Object Classes Challenge: A Retrospective". In: *International Journal of Computer Vision* 111.1 (2015), pp. 98–136. DOI: 10.1007/s11263-014-0733-5. URL: http://dx.doi.org/10.1007/s11263-014-0733-5.

[11] J. Friedman, R Tibshirani, and T. Hastie. *The Elements of Statistical Learning*. Springer Series in Statistics, 2001.

[12] Ross B. Girshick. "Fast R-CNN". In: *CoRR* abs/1504.08083 (2015). URL: http://arxiv.org/abs/1504.08083.

[13] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *CoRR* abs/1311.2524 (2013). URL: http://arxiv.org/abs/1311.2524.

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[15] Hayit Greenspan, Bram van Ginneken, and Ronald M. Summers. "Guest Editorial Deep Learning in Medical Imaging: Overview and Future Promise of an Exciting New Technique". In: *IEEE Transactions on Medical Imaging* 35.5 (May 2016), pp. 1153–1159. ISSN: 0278-0062. DOI: 10.1109/TMI.2016.2553401.

[16] Sebastian Gross, Stephan Palm, Jens J. W. Tischendorf, Alexander Behrens, Christian Trautwein, and Til Aach. "Automated classification of colon polyps in endoscopic image data". In: (2012). DOI: 10.1117/12.911177. URL: http://dx.doi.org/10.1117/12.911177.

[17] European Colorectal Cancer Screening Guidelines Working Group, L von Karsa, J Patnick, N Segnan, W Atkin, S Halloran, I Lansdorp-Vogelaar, N Malila, S Minozzi, S Moss, P Quirke, R J Steele, M Vieth, L Aabakken, L Altenhofen, et al. "European guidelines for quality assurance in colorectal cancer screening and diagnosis: Overview and introduction to the full Supplement publication". In: (2013). DOI: 10.1055/s-0032-1325997. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4482205/.

[18] M. Häfner, M. Liedlgruber, A. Uhl, A. Vácsei, and F. Wrba. "Delaunay triangulation-based pit density estimation for the classification of polyps in high-magnification chromo-colonoscopy". In: *Computer Methods and Programs in Biomedicine* 107.3 (2011), pp. 565–581. DOI: 10.1016/j.cmpb.2011.12.012. URL: http://dx.doi.org/10.1016/j.cmpb.2011.12.012.

[19] M. Häfner, M. Liedlgruber, A. Uhl, A. Vécsei, and F. Wrba. "Color treatment in endoscopic image classification using multi-scale local color vector patterns". In: *Medical Image Analysis* 16.1 (2011), pp. 75–86. DOI: 10.1016/j.media.2011.05.006. URL: http://dx.doi.org/10.1016/j.media.2011.05.006.

[20] Sae Hwang, JungHwan Oh, W. Tavanapong, J. Wong, and P.C. de Groen. "Polyp Detection in Colonoscopy Video using Elliptical Shape Feature". In: *Proc. of ICIP*. Sept. 2007, pp. 465–468.

[21] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167 (2015). URL: http://arxiv.org/abs/1502.03167.

[22] Michal F. Kaminski, Jaroslaw Regula, Ewa Kraszewska, Marcin Polkowski, Urszula Wojciechowska, Joanna Didkowska, Maria Zwierko, Maciej Rupinski, Marek P. Nowacki, and Eugeniusz Butruk. "Quality Indicators for Colonoscopy and the Risk of Interval Cancer". In: *New England Journal of Medicine* (2010). PMID: 20463339. DOI: 10.1056/NEJMoa0907667. URL: http://dx.doi.org/10.1056/NEJMoa0907667.

[23] J Kang and R Doraiswami. "Real-time image processing system for endoscopic applications". In: *Proc. of IEEE CCECE*. 2003.

[24] Kreftregisteret. *Tykk- og endetarmskreft*. Jan. 2016. URL: http://www.kreftregisteret.no/no/Generelt/Fakta-om-kreft-test/Tykk--og-endetarmskreft/.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[26] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* (May 2015), pp. 436–444. URL: http://dx.doi.org/10.1038/nature14539.

[27] Baopu Li and M.Q.-H. Meng. "Tumor Recognition in Wireless Capsule Endoscopy Images Using Textural Features and SVM-Based Feature Selection". In: *Trans. on ITBM* (May 2012).

[28] Mathias Lux, Michael Riegler, Pål Halvorsen, Konstantin Pogorelov, and Nektarios Anagnostopoulos. "LIRE: open source visual information retrieval". In: *Proc. of MMSys*. 2016.

[29] A.V. Mamonov, I.N. Figueiredo, P.N. Figueiredo, and Y.-H.R. Tsai. "Automated Polyp Detection in Colon Capsule Endoscopy". In: *Transs on MI* (2014).

[30] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. MIT press, 2012.

[31] Ruwan Nawarathna, JungHwan Oh, Jayantha Muthukudage, Wallapak Tavanapong, Johnny Wong, Piet C De Groen, and Shou Jiang Tang. "Abnormal image detection in endoscopy videos using a filter bank and local binary patterns". In: *NC* 144 (2014), pp. 70–91.

[32] Anh Nguyen, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *arXiv preprint arXiv:1412.1897* (2014).

[33] Jessica B. O'Connell, Melinda A. Maggard, and Clifford Y. Ko. "Colon Cancer Survival Rates With the New American Joint Committee on Cancer Sixth Edition Staging". In: *JNCI: Journal of the National Cancer Institute* (2004). DOI: 10.1093/jnci/djh275. URL: +%20http://dx.doi.org/10.1093/jnci/djh275.

[34] Sungheon Park, Myunggi Lee, and Nojun Kwak. "Polyp detection in colonoscopy videos using deeply-learned hierarchical features". In: *Seoul National University* (2015).

[35] Konstantin Pogorelov, Sigrun Losada Eskeland, Thomas de Lange, Carsten Griwodz, Kristin Ranheim Randel, Håkon Kvale Stensland, Duc-Tien Dang-Nguyen, Concetto Spampinato, Dag Johansen, Michael Riegler, and Pål Halvorsen. "A Holistic Multimedia System for Gastrointestinal Tract Disease Detection". In: *Proc. of MMSYS*. 2017.

[36] Konstantin Pogorelov, Kristin Ranheim Randel, Carsten Griwodz, Thomas de Lange, Sigrun Losada Eskeland, Dag Johansen, Concetto Spampinato, Duc-Tien Dang-Nguyen, Mathias Lux, Michael Riegler, and Pål Halvorsen. "Kvasir: A Multi-Class Image-Dataset for Computer Aided Gastrointestinal Disease Detection". In: *Proc. of MMSYS*. 2017.

[37] Konstantin Pogorelov, Kristin Ranheim Randel, Thomas de Lange, Sigrun Losada Eskeland, Dag Johansen, Carsten Griwodz, Concetto Spampinato, Mario Taschwer, Mathias Lux, Michael Riegler, and Pål Halvorsen. "Nerthus: A Bowel Preparation Quality Video Dataset". In: *Proc. of MMSYS*. 2017.

[38] Konstantin Pogorelov, Michael Riegler, Pål Halvorsen, Peter Thelin Schmidt, Carsten Griwodz, Dag Johansen, Sigrun L. Eskeland, and Thomas de Lange. "GPU-accelerated Real-time Gastrointestinal Diseases Detection". In: *Proc. of CBMS*. 2016.

[39] DMW Powers. "Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation". In: *Journal of Machine Learning Technologies* 2.1 (2011), pp. 37–63.

[40] Joseph Redmon. *Darknet: Open Source Neural Networks in C*. http://pjreddie.com/darknet/. 2013.

[41] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection". In: *arXiv preprint arXiv:1506.02640* (2015).

[42] Joseph Redmon and Ali Farhadi. "YOLO9000: Better, Faster, Stronger". In: *CoRR* abs/1612.08242 (2016). URL: http://arxiv.org/abs/1612.08242.

[43] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *CoRR* abs/1506.01497 (2015). URL: http://arxiv.org/abs/1506.01497.

[44] World Health Organization - International Agency for Research on Cancer. *Estimated Cancer Incidence, Mortality and Prevalence Worldwide*. 2012. URL: http://globocan.iarc.fr/Pages/fact_sheets_population.aspx.

[45] World Health Organization - International Agency for Research on Cancer. *Estimated Cancer Incidence, Mortality and Prevalence Worldwide*. 2012. URL: http://globocan.iarc.fr/Pages/fact_sheets_cancer.aspx?cancer=colorectal.

[46] Michael Riegler. "EIR - A Medical Multimedia System for Efficient Computer Aided Diagnosis". PhD thesis. University of Oslo, 2017.

[47] Michael Riegler, Mathias Lux, Carsten Griwodz, Concetto Spampinato, Thomas de Lange, Sigrun L Eskeland, Konstantin Pogorelov, Wallapak Tavanapong, Peter T Schmidt, Cathal Gurrin, et al. "Multimedia and Medicine: Teammates for Better Disease Detection and Survival". In: *Proc. of ACM MM*. 2016, pp. 968–977.

[48] Michael Riegler, Konstantin Pogorelov, Sigrun Losada Eskeland, Peter Thelin Schmidt, Zeno Albisser, Dag Johansen, Carsten Griwodz, Pål Halvorsen, and Thomas de Lange. "From Annotation to Computer Aided Diagnosis: Detailed Evaluation of a Medical Multimedia System". In: *Transactions on Multimedia Computing, Communications and Applications* 9.4 (2017).

[49] Michael Riegler, Konstantin Pogorelov, Pål Halvorsen, Thomas de Lange, Carsten Griwodz, Peter Thelin Schmidt, Sigrun L. Eskeland, and Dag Johansen. "EIR - Efficient computer aided diagnosis framework for gastrointestinal endoscopies". In: *Proc. of CBMI*. 2016. DOI: 10.1109/CBMI.2016.7500257.

[50] Michael Riegler, Konstantin Pogorelov, Jonas Markussen, Mathias Lux, Håkon Kvale Stensland, Thomas de Lange, Carsten Griwodz, Pål Halvorsen, Dag Johansen, Peter T Schmidt, and Sigrun L. Eskeland. "Computer Aided Disease Detection System for Gastrointestinal Examinations". In: *Proc. of MMSys*. 2016.

[51] Rishi Kumar Samer Hijazi and Chris Rowen. "Using Convolutional Neural Networks for Image Recognition". In: (2015). URL: https://ip.cadence.com/uploads/901/cnn_wp-pdf.

[52] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2014). URL: http://arxiv.org/abs/1409.1556.

[53] Ingo Steinwart and Andreas Christman. *Support Vector Machines*. Information Science and Statistics, 2008.

[54] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. "Rethinking the Inception Architecture for Computer Vision". In: *CoRR* abs/1512.00567 (2015). URL: http://arxiv.org/abs/1512.00567.

[55] Nima Tajbakhsh, Suryakanth R Gurudu, and Jianming Liang. "Automatic polyp detection in colonoscopy videos using an ensemble of convolutional neural networks". In: *Proc. of IEEE ISBI*. 2015.

[56] Nima Tajbakhsh, Suryakanth Gurudu, and Jianming Liang. "Automated Polyp Detection in Colonoscopy Videos Using Shape and Context Information". In: *IEEE Transactions on Medical Imaging* 35.2 (Feb. 2016), pp. 630–644.

[57] Dayong Wang, Aditya Khosla, Rishab Gargeya, Humayun Irshad, and Andrew H. Beck. "Deep Learning for Identifying Metastatic Breast Cancer". In: (2016). URL: https://arxiv.org/abs/1606.05718v1.

[58] Y. Wang, W. Tavanapong, J. Wong, J. Oh, and P. C. de Groen. "Part-Based Multiderivative Edge Cross-Sectional Profiles for Polyp Detection in Colonoscopy". In: *IEEE Journal of Biomedical and Health Informatics* 18.4 (June 2014), pp. 1379–1389. ISSN: 2168-2194. DOI: 10.1109/JBHI.2013.2285230.

[59] Yi Wang, Wallapak Tavanapong, Johnny S Wong, JungHwan Oh, and Piet C de Groen. "Detection of quality visualization of appendiceal orifices using local edge cross-section profile features and near pause detection". In: *BME* (2010).

[60] Yi Wang, Wallapak Tavanapong, Johnny Wong, Jung Hwan Oh, and Piet C de Groen. "Polyp-Alert: Near Real-time Feedback during Colonoscopy". In: *Computer methods and programs in biomedicine* 120.3 (2015), pp. 164–179.

[61] Yi Wang, Wallapak Tavanapong, Johnny Wong, JungHwan Oh, and Piet C de Groen. "Computer-aided detection of retroflexion in colonoscopy". In: *Proc. of IEEE CBMS*. 2011, pp. 1–6.

[62] Yi Wang, Wallapak Tavanapong, Johnny Wong, JungHwan Oh, and Piet C de Groen. "Near Real-Time Retroflexion Detection in Colonoscopy". In: *IEEE Journal of Biomedical and Health Informatics* 17.1 (2013), pp. 143–152.

[63] Yi Wang, Wallapak Tavanapong, Johnson Wong, JungHwan Oh, and Piet C de Groen. "Part-Based Multiderivative Edge Cross-Sectional Profiles for Polyp Detection in Colonoscopy". In: *Journal of BMHI* 18.4 (2014), pp. 1379–1389.

[64] Wikipedia. *Convolutional neural network - Wikipedia*. https://en.wikipedia.org/wiki/Convolutional_neural_network. (Accessed on 02/09/2017). Feb. 2017.

[65] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *CoRR* abs/1311.2901 (2013). URL: http://arxiv.org/abs/1311.2901.

[66] Mingda Zhou, Guanqun Bao, Yishuang Geng, B. Alkandari, and Xiaoxi Li. "Polyp detection and radius measurement in small intestine using video capsule endoscopy". In: *Proc. of BMEI*. Oct. 2014, pp. 237–241.