Mimir: An Automatic Reporting and Reasoning System for Screening of the Gastrointestinal Tract Using Deep Neural Networks

Steven A. Hicks

Thesis submitted for the degree of
Master in Programming and Networks
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2018

Mimir: An Automatic Reporting and Reasoning System for Screening of the Gastrointestinal Tract Using Deep Neural Networks

Steven A. Hicks

Mimir: An Automatic Reporting and Reasoning System for Screening of the Gastrointestinal Tract Using Deep Neural Networks

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

Data is arguably one of the most valuable resources available today. More than ever, data is collected on such a large scale that we do not have the capacity to process it efficiently. In healthcare alone, there is an estimated 162 exabyte of data throughout the world, which is growing at the speed of approximately 2.5 exabytes per year [18]. Medical data in and of itself can be used for many things, such as patient follow-ups or recommendations. Nevertheless, to enable the use of this information to its fullest potential, we need sophisticated data analysis methods such as statistics or machine learning. Machine learning is a field where machines learn from data without explicitly being programmed. This process is often applied through supervised learning (machines learning from labeled data), unsupervised learning (machines learning from unlabeled data), or semi-supervised (machines learning from a combination of labeled and unlabeled data). Over the past few years, this field has been dominated by a growing class of algorithms known as deep learning. Inspired by the neurological connections in the animal brain, deep learning has made immense strides in the production of state-of-the-art results within many areas of data analytics [4]. Nowadays, deep learning based methods have become a popular topic within the medical field as well [7]. This has brought up some specific challenges which may make the application of these methods difficult, such as the lack of data or poor understanding of their internal workings. The latter issue, namely that deep learning is something of a "black box", is one of the biggest hurdles since it hinders the application of deep learning from being used in hospitals due to lack of trust and understanding. For this reason, we developed a medical reporting system, which focuses on transparency and understanding of its internal processes. In this thesis, we present this system and show how it may aid us in the development and understanding of deep neural networks.

Acknowledgements

I'd like to thank EVERYONE!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

The medical scenario of focus for this thesis will be on the field of gastrointestinal (GI) endoscopy, which in layman's terms is the conventional method of screening the digestive system through the use of a special type of camera. The digestive system is one of the most diverse and complex organ systems in the human body. With the sole responsibility of breaking down food into nutrients, it plays a pivotal role in the growth and development of any living person. However, this system is prone to many diseases ranging from minor annoyances to potentially life-threatening illnesses. In the GI tract alone, three of the six most common cancer types are found, and with an annual detection rate of 2.8 million new cancer cases, and a five-year mortality rate of 65%, this area is in much need for improvement. Early detection is vital for patient survival, but a standard issue among GI cancer types is that they exhibit little to no apparent symptoms before its too late. The current best working method for screening the GI tract for abnormalities is through endoscopy examinations, where one must rely on the doctor's ability to detect early signs of cancer in the form of its precursors (polyps, which are abnormal tissue growths often taking the shape of a mushroom). This has proven to be an issue in and of itself, where the doctor's ability to detect polyps has shown to be a more important predictor than that of the most common risk factors associated with the diagnosis of this disease [11].

Looking to improve the state of GI disease detection, one must first have a metric to measure them by. In the field of GI endoscopy, this is commonly done through manually written documentation of the performed procedures. This documentation is essential, as it might be the only evidence of a procedure taking place. Despite the introduction of various standards, such as colonoscopy reporting and data system (CO-RADS) and Minimal Standard Terminology (MST), documentation of performed endoscopies is generally poor, often being submitted incomplete and without the use of standardized language. Reports attributed this to a general lack of train-

ing and knowledge around the beforementioned guidelines and mentioned that the use of computerized systems in the form of endoscopic electronic medical records (EEMRs) would most likely improve this field.

In the last few years, a rising trend of using deep learning based methods has emerged, having seen much success in various fields including medicine [75]. Automatic detection of disease could be of great help in lowering the misrate of abnormalities (polyps and other illness) when screening the GI tract. Additionally, automatic detection of notable findings in the digestive system, such as anatomical landmarks and polypectomy markings (surgical markings for polyp removal), could be of great aid in the generation of documentation, as these findings mark important information that should be part of any endoscopy report. Although these methods have shown to work well within the medical domain [26, 60, 69], often showing improved results over their traditional counterparts, there is one aspect of deep learning which makes it difficult to implement in real-world practice. Neural networks are often considered to be a "black box" because the internal process which leads to a specific result is neither easy to understand nor easy to interpret. This poor understanding has led to a lack of trust in these systems, often leading to medical experts favoring traditional based methods, even though they are often less accurate than their deep learning counterparts.

With the performance and complexity of deep learning based neural networks steadily increasing, we see that they have much potential in aiding medical doctors in the detection of severe disease. However, the lack of understanding and trust is concerning. Opening this "black box" through the use of modern interpretability methods would not only aid in the building of trust and understanding among medical experts but could also be used to produce quality endoscopy reports. These open questions motivated our research into the field of deep learning interpretability and automatic generation of quality endoscopy documentation.

1.2 Problem Statement

Based on the background and motivation presented in the previous section, we decided to look into improving the area of deep learning understanding and transparency. We see that this is an important piece in building trust and increasing the general acceptance of these algorithms. Additionally, by providing detailed explanations into why and how a model provides a given result, we may be able to use this information in the production of complete and standard compliant endoscopy reports. As for the scope of this thesis, we will be focusing on the completion of three main objectives, which act as the initial steps of completing this overarching goal. The three objectives of this thesis are as follows:

Objective 1 Research and develop a system which gives non-technical users a better understanding of why a neural network presents

a given result. This system should be aimed at medical doctors conducting examinations and documentation abnormalities found in the GI tract.

Objective 2 Provide a proof-of-concept implementation of automatic GI report generation based on the findings of automatic analysis done through the use of a deep neural network.

Objective 3 Use various visualization techniques to get a better understanding of the internal working of a deep neural network. This newly gained knowledge should be used in the development of preprocessing steps with the purpose of training quality and robust analytical models based on deep learning.

As part of the three objectives require the research and development of a system. We also decided to define three requirements which we would keep in mind when developing the initial prototype of our automatic reporting system with a focus on transparency and understanding. The three system requirements are as follows:

Requirement 1 The system should give non-technical users the ability to understand why a neural network based model suggest a given disease diagnosis.

Requirement 2 The system should provide tools for medical documentation and suggest image attachments based on the analysis done by the underlying analytical model.

Requirement 3 The system should be able to aid in the development and improvement of deep learning based models and datasets.

With our research objectives in place, we started development on a system which would meet our stated system requirements. This system would then aid us in the answering of our previously defined research objectives.

1.3 Limitations

Based on the research question and its objectives, the scope of this thesis is researching and developing an automatic reporting system with a focus on deep neural network understanding and transparency for use in the medical domain. As a first use case, we will be applying this system to analysis, detection, and documentation of the anatomy and diseases found in the GI tract. We have limited ourselves to eight different classes due to two primary constraints. Firstly, there are far too many parts and diseases found in the GI tract, so keeping the number of classes to a manageable number is essential considering our time constraint. Secondly, and most

important, there is a lack of publicly available medical data, making it difficult to be picky when it comes to which medical disease and anatomical parts we wish to analyze. This lack of medical data is also the reason for the selection of the eight classes used for classification, as we will be using the publicly available Kvasir (v2) [76] dataset to train and evaluate our analytical models. The eight included classes are as follows; Ulcerative Colitis, Esophagitis, Polyps, Cecum, Z-line, Pylorus, Dyed lifted Polyps and Dyed Resection margins. Running and verifying the developed system on further diseases or other application scenarios is out of the scope of this thesis.

Considering the scope of this thises, we will limit ourselves to focusing on image classification with the use of deep convolutional neural networks (CNNs). Although there are other methods commonly used within this field, e.g., manual feature extraction, recurrent neural networks (RNNs), etc., we have chosen CNNs as they are currently the most popular methods of autoamtic image classification.

1.4 Research Methods

Research can be performed in a variety of ways. For this thesis, we have decided to use Association for Computing Machinerys (ACMs) research methodology. In 1989, the ACM Education Board assigned a task force to compile the core fundamentals of computer science and computer engineering into a detailed report [21]. The report describes the discipline of computing as being split between three paradigms; (i) theory, (ii) abstraction, and (iii) design. The work conducted over the course of this thesis touches upon each of these paradigms in a variety of ways. Below, we give a brief description of each paradigm and discuss how our work fits into each of them.

1.4.1 Theory

The "theory" paradigm is rooted in mathematics and relates to the development of a coherent and valid theory. The report describes this phase as being made up for four steps, which are described as follows; (i) characterize objects of study (definition), (ii) hypothesize possible relationships among them (theorem), (iii) determine whether the relationships are true (proof), and (iv) interpret results.

This paradigm is supported by the analyzed relationship between the neural networks feature activations and its predicted output. Using this information, we applied various pre-processing techniques to the training data and reran the same analysis, interpreting how the change in dataset affected the change in class scores.

1.4.2 Abstraction

The "abstraction" paradigm is rooted in the experimental scientific method and relates the investigation of a phenomenon, e.g., hypothesis. The report describes this phase as a process consisting of four steps, which are described as follows; (i) form a hypothesis, (ii) construct a model and make a prediction, (iii) design an experiment and collect data, and (iv) analyze results.

This paradigm is supported by our experiments performed on the various models trained over the course of this thesis. Based on the predictions of a given neural network based model, we developed a hypothesis on why we thought a given model assigned a particular class with a probability. This hypothesis was then put to the test using our developed system, of which we were able to either verify or refute our original theory.

1.4.3 Design

The "design" paradigm is closely related to engineering and relates to the construction of a system, e.g., software, hardware, etc. The report describes this phases as a process consisting of four steps, which are described as follows; (i) state requirements, (ii) state specifications, (iii) design and implement the system, and (iv) test the system.

Our work supports this paradigm through the implementation of a prototypical system called Mimir [46]. This system was used as part of this thesis to conduct a variety of experiments which proved to show the usefulness of the system.

1.5 Main Contributions

Over the course of this thesis, we researched and developed a system for automatic detection and reporting of disease found in the GI tract called Mimir [45, 46]. This system focused on making the analysis performed by the underlying neural network transparent and understandable through a series of intermediate visualizations, which purpose was to further increase the acceptance and trust. As defined in our problem statement (Section 1.2), we set three system requirements which our system should meet to be considered complete (within the context of this thesis). The following reiterates the requirements and describes what how our system meets them:

Requirement 1 *The system should give non-technical users the ability to understand why a neural network based model suggested a given disease diagnosis.*

This requirement is supported by the neural network dissection tool as part of Mimir, which generate visualizations based on what the neural network "sees" when making a given prediction.

Requirement 2 *The system should provide tools for medical documentation and suggest image attachments based on the analysis done by the underlying analytical model.*

This requirement is supported by the report generation tool, which suggests the most relevant images from the automatic analysis done by the underlying deep neural network.

Requirement 3 *The system should be able to aid in the development and improvement of deep learning based models and datasets.*

This requirement is supported by the neural network dissection tool, which in addition to providing insight into the analysis of a deep neural network, also provides tools for uploading and managing various deep learning based models.

With these system requirements fulfilled, we look at how Mimir solves our three research objectives which define what work should have been done over the course of this thesis:

Objective 1 *Research and develop a system which gives non-technical users a better understanding of why a neural network presents a given result. This system should be aimed at medical doctors conducting examinations and documentation abnormalities found in the GI tract.*

This objective is supported by the development of Mimir, which provides a tool for dissecting the internal layers of a deep CNN. Using this tool, a doctor may verify that the diagnosis suggested by the system is in fact due to the detection of said disease, and not due to artifacts or noise commonly found in medical images.

Objective 2 *Provide a proof-of-concept implementation of automatic GI report generation based on the findings of automatic analysis done through the use of a deep neural network.*

This objective is supported by the report generation tool included in Mimir, which suggests relevant images based on a diagnosis proposed by the system. As stated in the objective, this is currently a proof-of-concept, meaning it is expected to be improved through future work.

Objective 3 *Use various visualization techniques to get a better understanding of the internal working of a deep neural network. This newly gained knowledge should be used in the development of pre-processing steps with the purpose of training quality and robust analytical models based on deep learning.*

This last objective is supported by our use of Mimir to analyze five neural network based models, each using a different standardized architecture, with the purpose of finding faults in its training. Based on the performed analysis, we derived two pre-processing steps applied to Kvasir (v2) [76] dataset, which showed to improve the classification score of all models except one. This objective is also supported by a published paper [44], where we showcase part of the experiments conducted over the course of this thesis.

Through the work produced in this thesis, we learned that using neural network based visualizations may provide sufficient knowledge into what pre-processing steps may lead to improved classification performance. Specifically, we improved the performance of a deep neural network trained to detect disease and anatomy of the GI tract.

Each objective is supported by published papers, each paper can be seen in Appendix B, where Paper B.1 [46] and Paper B.2 [45] relate to the first two objectives, and Paper B.3 [44] relates to the last objective.

1.6 Thesis Outline

This thesis is split into five chapters, with the first two being introductory and filling in the necessary background to fully understand the rest of the thesis. Chapters 3 and 4 describe the work done over the curse of this thesis, with accompanied published papers for both chapters located in the appendix. The last chapter is the conclusive chapter, which sums up the produced results and presents the future work. Below we have included a summary of each chapter (excluding chapter 1).

Chapter 2: Deep Learning and Automatic Reporting for Medical Multimedia
We present the medical and technical background of using deep learning methods in the medical domain, specifically gastroenterology. The overall structure of this chapter is mainly split into two main parts, one concerning the medical background, the other regarding the technical details of deep learning. For the medical background, we present the background to the current state of endoscopic disease detection and documentation through endoscopic reports. This includes a look at various parts of the GI anatomy and the various diseases commonly found there, with a more detailed look at the eight classes used for classification. We also look at the current state-of-the-art methods of GI examinations in the form of different types of endoscopy. Lastly, we dive into the current state of endoscopic reporting, where we look at its present faults and potential ways it can be improved.

For the more technical deep learning part, we start with the very basics, explaining what makes up a traditional neural network. We then expand on this information by introducing more complex networks in the form of CNNs and describe the various aspects of these networks

which make them perform so well for image classification tasks. With a basic understanding of traditional neural networks and their more complex extension in the form of CNNs, we discuss current problems with applying these methods to mission-critical domains, specifically medicine. Lastly, we look at various methods of trying to get some understanding of how these methods work, and what this may tell us about their inner workings.

Chapter 3: Mimir: An Automatic Reporting System for Endoscopic Examinations
We present the automatic reporting system developed over the course of this thesis. This chapter looks at the technical implementation of the system, explaining how, what and why we use certain technologies. We present our system through a detailed guide on how to use its various included tools and suggest potential use case scenarios which we expect would be a good fit for this system. This chapter directly relates to our research objectives 1 and 2, as stated in our problem statement (Section 1.2).

Chapter 4: Case Study on Mimir for use in Classification Understanding
We present the experiments conducted to gain a better understanding of deep neural networks trained on medical image data, specifically Kvasir (v2). This includes a brief description of the various architectures and datasets used for training, how we performed training and evaluation and how we conducted our analysis of each model. We then present our results through evaluation metrics and have a look at how some of the previous visualizations changed after training with the two derived datasets based on Kvasir (v2) [76]. This chapter directly relates to objective 3 stated in our problem statement (Section 1.2).

Chapter 5: Conclusion and Further Work
Finally, we conclude this thesis with a summary of what we have presented, a discussion on potential future work.

Chapter 2

Deep Learning and Automatic Reporting for Medical Multimedia

In recent years, deep learning has shown to improve on the state-of-the-art in many fields such as object recognition, language translation, and robotics. In addition to this, there has been much progress in applying these methods to the field of medicine as well, where deep neural networks have successfully aided in the diagnosis of brain disease, skin cancer, and also used as a risk assessment tool for breast cancer patients [27, 60, 62].

In this chapter, we present the necessary background and related works of applying deep learning methods to the field of GI disease detection and diagnosis. This will primarily be covered over the course of two parts, one covering the necessary medical background and the other looking at the technical use of deep learning in mission-critical fields such as the medical domain. The theory and research presented in this chapter was part of the initial work done to successfully fulfill our three research objectives stated in Section 1.2.

We begin with a case study on the GI tract, where we start by giving a short introduction to the purpose of this organ system and how it aids the human body through the digestion of food. With a good understanding of the GI anatomy, we look to the current state-of-the-art methods of GI disease detection through the use of various types of endoscopies. We then present a detailed look at the eight classes which will later be used for training and classification. As defined by our research objective, we look at the current state of GI reporting, reviewing various studies conducted within this areas, and discuss how this field may be improved.

The second part will focus on machine learning, with on deep learning, which in includes various architectures, applications, and different methods of interpreting their inner workings and output. We start by presenting the very basics of a traditional neural network, explaining a simple Multilayer Perceptron (MLP) from the ground up. This will give some in-

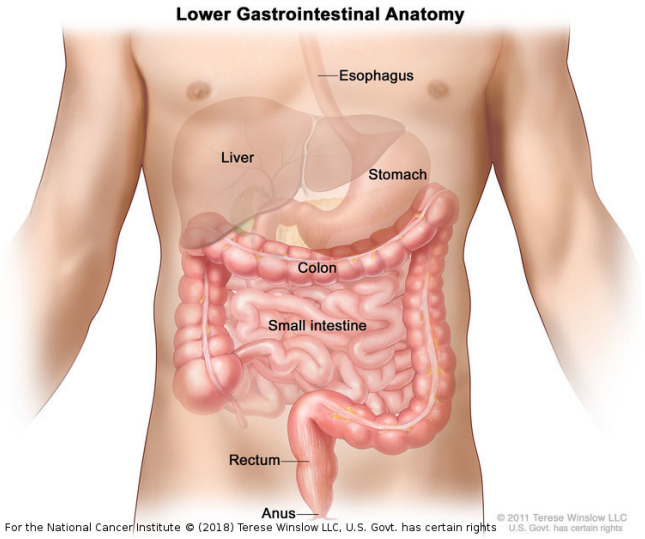
tuition of the basic structure of a typical neural network and the various algorithms used to train them. We then move on to a more advanced architecture, CNNs, where we cover the unique attributes that make them specifically tailored for image classification. This should give the necessary background to fully understand how we use CNNs to analyze medical image data using the neural network dissection tool part of Mimir, and the various experiments conducted over the course of this thesis. Additionally, we look at multiple methods of applying deep neural networks to the field of medicine, covering some of the successes and challenges of utilizing these methods in this domain. Lastly, we cover the various methods of trying to gain some understanding of how the internal processes of a deep neural network produces its results, and how this knowledge may help both researchers and end-users alike.

2.1 Case Study on Detection and Documentation of Disease in the Gastrointestinal Tract

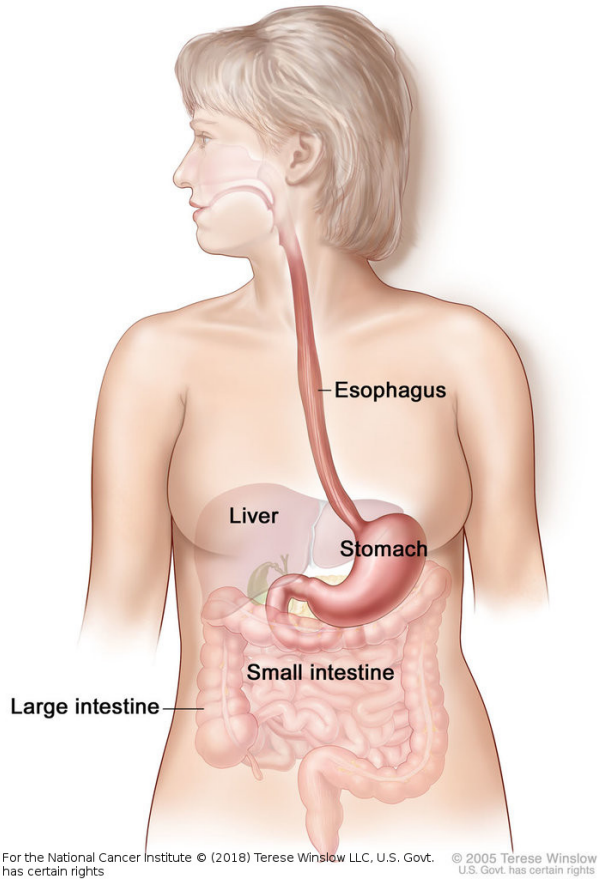
The GI tract, sometimes referred to as the digestive tract, is the primary organ of the human digestive system. Along with various accessory organs (tongue, liver, pancreas, etc.), its main function is to intake food, absorb nutrients through digestion, and dispose of it through feces or urine. As an initial use case, we limit our work to focus on the detection and documentation of eight distinct anatomical parts (including abnormalities and polypectomy markings) of the GI anatomy. The eight classes are divided into three categories; abnormalities (3), anatomical landmarks (3) and polypectomy markings (2). The reason behind this limitation is quite simply the general lack of annotated image data available for public use. We, therefore, decided to focus on the image classes part of the Kvasir (v2) [76] dataset, as this is a publicly available dataset.

2.1.1 The Gastrointestinal Tract

As we briefly mentioned above, the main purpose of the GI tract is to absorb nutrients through the digestion of food and dispose of it through waste. We generally draw a distinction between the lower and upper parts of the GI anatomy, with the upper GI tract spanning from the mouth to the ileum, and lower GI tract spanning from the cecum to the anus. It is worth noting that some make a third distinction by denoting the small intestine as the middle GI tract, the reason for this distinction is that the two procedures used to inspect the upper and lower GI tract generally do not cover the small intestine, which requires a more extensive operation through capsule endoscopy or enteroscopy. As for this work, we will keep it simple by using the first division. An illustration of the upper and lower GI tract in seen in Figure 2.1, which will be useful when we next give a brief description of the digestion process of the GI tract.



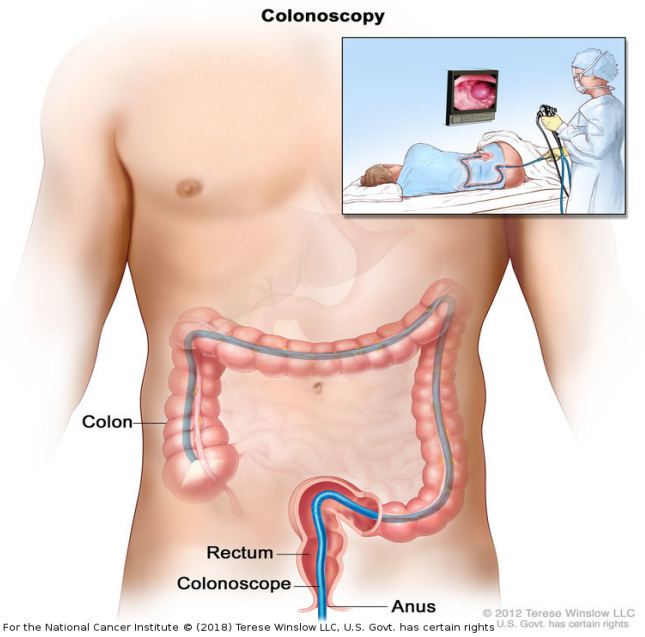
(a) An image showing what is commonly considered the lower gastrointestinal tract.



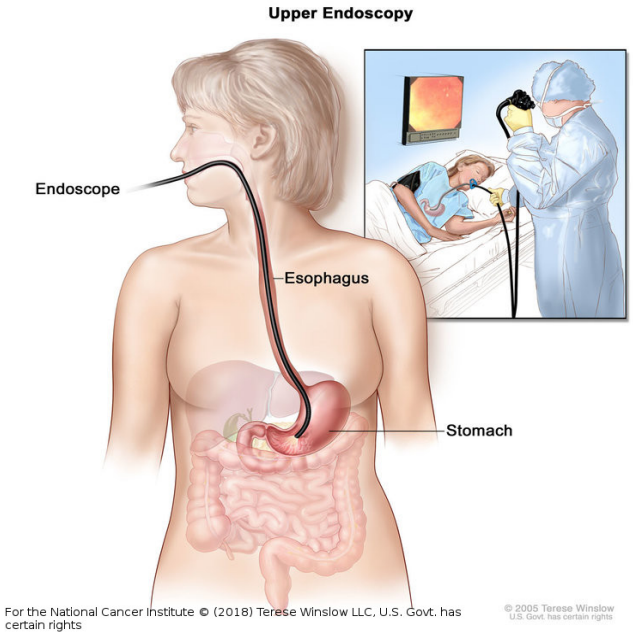
(b) An image showing what is commonly considered the upper gastrointestinal tract.

Figure 2.1: Two illustrations covering the lower (2.1a) and upper (2.1b) gastrointestinal tract.

The digestive process starts at the oral cavity (mouth), where food is inserted and passed along a hollow-like tube, called the esophagus, which leads into the stomach. Here the food is mixed together and broken down by acids and enzymes before being passed into the duodenum (the first part of the small intestine). The small intestine consists of duodenum, jejunum, and ileum; and is where the majority of nutrient absorption takes place. Lining the walls of the small intestine is a mucosal membrane, or mucosa, which secretes enzymes and bile salts from the pancreas and gallbladder to further break down and digest the partially digested food received from the stomach, which in turn is absorbed by the bloodstream. Whats left over is passed into the initial part of the large intestine (colon). The large intestine consists of the appendix, cecum, ascending, transverse, descending colon, sigmoid colon, and rectum; and is responsible for absorbing the remaining water, salts, sugars and vitamins from the indigestible food. It ends at the anus, where the remaining food is expelled in the form of feces.



(a) An image showing the extent of a colonoscopy, note that the examination starts at the rectum and ends at the entrance to the small intestine.



(b) An image showing the extent of a gastroscopy, note that the examination starts at the mouth and ends at the stomach.

Figure 2.2: Two illustrations showing two conventional forms of endoscopy, colonoscopy and gastroscopy.

The GI tract may be home to a multitude of disease, including infection, inflammation, and cancer. Colorectal cancer (CRC) is a severe disease that makes up approximately 10% of total cancer cases [28]. A common problem with CRC is that it generally does not exhibit any apparent symptoms before it is too late. Therefore, it is crucial that the GI tract is routinely screened for disease and CRCs precursors in the form of polyps. The current state-of-the-art method of screening the GI tract is through various types of endoscopy, which we describe in further detail below.

2.1.2 Gastrointestinal Endoscopy

GI endoscopy is a procedure where the GI tract is examined through the use of an endoscope for detection of abnormalities in the form of disease, infection or other special conditions. Unlike other medical imaging techniques (x-ray, ultrasound, etc.), endoscopes are inserted directly into the organ to be examined. This is done by inserting a long flexible tube attached with a small camera into either the mouth (gastroscopy) or anus (colonoscopy). This is shown in Figure 2.2, where we see Figure 2.2a is of a colonoscopy and Figure 2.2b is of a gastroscopy. The overall procedure is considered to be safe, but complications do happen and in severe cases may be life-threatening. There is a variety of literature on the subject of endoscopy complications, with different studies presenting

slightly varied results. But the overall consensus is that complications of any kind occur in less than 2% of all endoscopies, with life-threatening complications occurring in well under 1% [33, 37, 78]. The risk of endoscopies vary depending on certain risk factors, and what procedures are performed under examination (such as polypectomy). Common complications include perforation (tear in the gut wall), a reaction to the sedation, infection, bleeding, and pancreatitis as a result of endoscopic retrograde cholangiopancreatography (ECRP).

For a lot of people, endoscopies are expensive, invasive, and the cause of high anxiety and discomfort. A single endoscopic procedure (colonoscopy or gastroscopy) averages at about 3000 U.S. dollars [85], making it a significant investment for a sizable part of the U.S. population. This may cause patients to forego treatment as they can not justify the cost. Anxiety is also a large barrier between patients and the surgical room. A recent study found that most patients are more anxious about the colonoscopy procedure itself, with factors including no previous colonoscopies and confusing instructions [93]. This shows that once a patient has undergone an endoscopic procedure, he/she is more likely to continue following the recommended five-year routine (looking past the variable of cost).

Endoscopies are also quite time demanding, requiring about one medical-doctor-hour and two nurse-hours [59], not including reporting and eventual follow-ups, and therefore do not scale well to a large population. Also, a recent study showed that about 17% of patients diagnosed with CRC had been investigated in the previous three years [102]. With the typical adenomatous polyp taking about five to ten years to become malignant [95], we can conclude that standard colonoscopies have a high miss rate due to the endoscopists inability to detect polyps. This is often referred to as the post-colonoscopy colorectal cancer (PCCRC) rate and is a key quality indicator of the performance of colonoscopies.

In addition to conventional colonoscopies, a relatively new endoscopic procedure using camera attached pills has been put to use in the last 20 years. This procedure is commonly called a video capsule endoscopy (VCE) and may be a solution to the scalability and cost inefficiencies of conventional endoscopy. However, in its current state, it is mostly used as an additional method used if conventional endoscopies do find any abnormalities when the patient is still showing signs of potential GI disease. Additionally, it is also used as an alternative method of inspecting the small intestine, as the conventional method of examining this organ has a higher risk of complications than that of a standard colonoscopy or endoscopy [36]. In the next section, we will describe VCE in more detail.

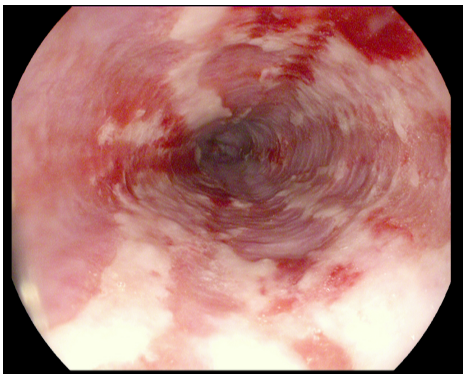
Figure 2.3: A video capsular endoscopy pill on its way into a patients mouth.

2.1.3 Wireless Video Capsular Endoscopy

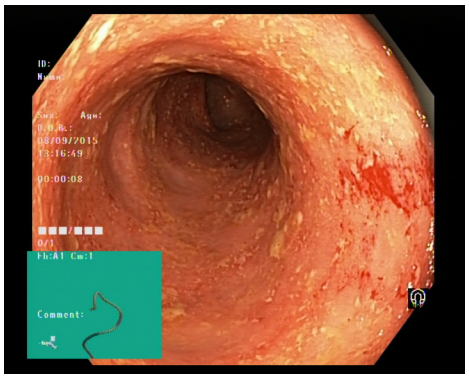
As mentioned in the previous section, the current state of conventional endoscopy does not scale well to a large population because of its high costs, time requirements and lack of qualified medical personnel. A proposed solution to these problems is the usage of a VCE. A VCE is a small camera placed in a vitamin-sized capsule which is inserted at the mouth and travels through the GI tract. Figure 2.3 shows an example of such a pill being swallowed by a patient. The capsule is outfitted with various devices such as image sensors, bleeding sensors, pH-sensors, antennas, batteries, light sources and wireless transceivers. The small capsule travels through the GI tract, taking images of the mucosa and transmits them to an external transceiver.

The idea here is that when it is time for an examination, the patient purchases a VCE capsule at their local pharmacy. They synchronize the pill to a wireless device such as a phone to receive a stream from the video capsule going through GI tract. This can be in the form of images or messages (such as notifying that it has reached a particular anatomical landmark). It is estimated to take about 10 to 12 hours for a procedure, and about 24 to 72 hours to be excreted [104]. When the pill has passed through the digestive system, the receiving device relays the data to a server where analysis is performed. From there, a trained professional can look at the results and verify that the analysis has been performed correctly and decide the appropriate next step. This is a bit out of scope for the course of this thesis, but it may play a pivotal role in the automatic documentation of endoscopy reports.

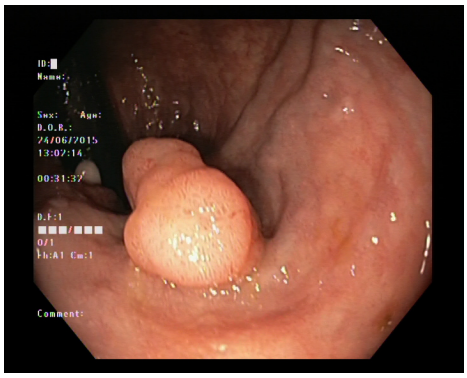
With the various methods of screening the GI tract covered, it is time to take a closer look at different parts of the GI anatomy. This



(a) Image of an esophagus inflicted by esophagitis. Notice the the red markings on the wall of the esophagus.



(b) Image of the colon mucosa severly inflicted by ulcerative colitis.



(c) Image of a adenomatous polyp located inside the colon.

Figure 2.4: Sample images from each of the three classes of abnormalaites as part of (and taken from) the Kvasir (v2) dataset.

includes multiple abnormalities, anatomical landmarks, and surgical polyp markings. The covered parts of the GI tract in the upcoming three chapters were chosen because of the focus on classifying images related to the described findings. This will become clearer under Chapter 3.

2.1.4 Abnormalities and Disease Found in the Gastrointestinal Tract

Gastrointestinal disease is generally split between three areas of the GI tract; Esophagus disease, gastric disease, and intestinal disease. Esophagus disease includes various abnormalities and disorders affecting the esophagus (between the mouth and stomach). Common disease found here includes gastroesophageal reflux disease (GERD), Barrett's esophagus, esophagitis, and Boerhaave syndrome. Gastric disease includes disease found in the stomach; this includes gastritis, gastroparesis, and various cancers. Intestinal disease covers the disease found in the small and

large intestine, and include ulcerative colitis, colon polyps, and coeliac disease. Note that disease affecting the oral cavity (mouth) are generally not included when referring to disease of the GI tract, albeit some disease found here might be the cause of a GI disease such as GERD, which can cause acid erosion of the teeth or halitosis (bad breath). In the upcoming few sections, we give a detailed look at the disease which is part of the automatic reporting system and can be seen in Figure 2.12.

2.1.4.1 Esophagitis

Esophagitis is an inflammation, irritation or swelling of the esophagus. This is often caused by gastric acid passing back up the esophagus (often a result of GERD), vomiting or hernias. An example can be seen in Figure 2.4a, where we see a highly inflamed esophagus denoted by the red markings on the wall of the mucosa. Detection is important for proper treatment and to prevent further irritation. Most patients improve over the course of two to four weeks depending on the severity of inflammation. The severity of esophagitis can generally be categorized into four grades depending on the measured breaks in the mucosa (an area of slough or erythema which causes a demarcation between it and the mucosa), each with increasing severity (grades taken from [66]):

Grade A: One (or more) mucosal break no longer than 5 mm, that does not extend between the tops of two mucosal folds.

Grade B: One (or more) mucosal break more than 5 mm long that does not extend between the tops of two mucosal folds.

Grade C: One (or more) mucosal break that is continuous between the tops of two or more mucosal folds but which involves less than 75% of the circumference.

Grade D: One (or more) mucosal break which involves at least 75% of the esophageal circumference.

Recent research has provided a treatment method involving surgically placing a ring of magnetic titanium beads near the lower esophageal sphincter. The procedure is called magnetic sphincter augmentation device (MASD) and has shown vast improvements with 70% of patients achieving normalized esophageal pH levels [89].

2.1.4.2 Ulcerative Colitis

Ulcerative colitis is a chronic inflammatory disease which affects the colon (large intestine) and rectum. The disease usually begins to develop before the age of 30 and is most commonly found towards the lower section of the large intestine (sigmoid colon) and rectum, but can affect the entire colon.

Primary symptoms of this disease include abdominal pain, cramping, and diarrhea mixed with blood, secondary symptoms include weight loss, fever and anemia [1]. The exact cause of ulcerative colitis is still unknown, with doctors speculating that the immune system overreacts on normal bacteria found in the digestive tract [24]. The disease can cause long-lasting inflammation and ulcers in the GI tract. Depending on the severity, it can be quite uncomfortable and may eventually become life-threatening.

2.1.4.3 Polyps

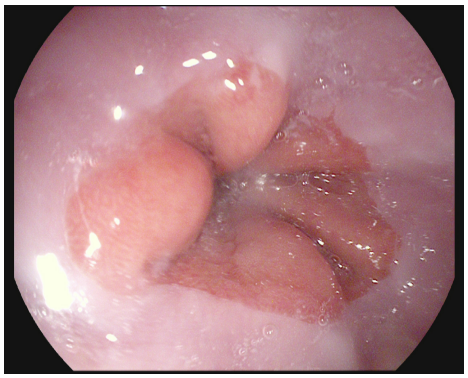
Colon polyps are small outgrowths from the mucosa and are either flat, elevated or pedunculated (connected to a thin stalk). They are formed when mutations in certain genes begin to divide, even though new cells are not needed. The result of this is a clump of cells, which in its basic form is referred to as a polyp. We typically divide polyps into two categories, non-neoplastic and neoplastic. Non-neoplastic polyps include hyperplastic polyps, inflammatory polyps, and hamartomatous polyps. These are normally new formations and have little chance of becoming cancerous. Neoplastic polyps include serrated and adenomatous polyps, of which, serrated polyps have a higher chance of being malignant, but adenomatous polyps may become cancerous as well. Figure 2.4c shows an example of an adenomatous polyp, located approximately in the middle of the image.

A general rule of thumb is that the bigger the polyp is, the more likely it is to become malignant. As all polyps have a chance of becoming cancerous over time, they are always removed even though they pose little threat to the patient at the time of removal. It is therefore vital that polyps are detected and removed before they reach a dangerous state. Polyps usually do not exhibit any external symptoms. It is therefore essential to have regular screenings to have them removed as early as possible.

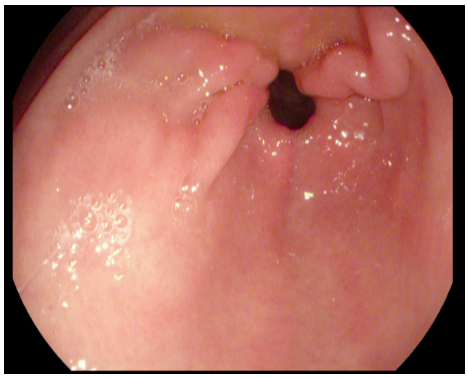
2.1.5 Anatomical Landmarks

Anatomical landmarks are used as a reference point to determine how far the endoscopic device (conventional or VCE) has made it into the colon or esophagus. This reference point is used to determine the location of potential findings and as an indication of a completed endoscopy. Additionally, some disease tends to infect the surrounding area of certain anatomical landmarks, such as GERD, which is commonly diagnosed through inspection of the z-line. Examples of the following described landmarks can be seen in figure 2.5.

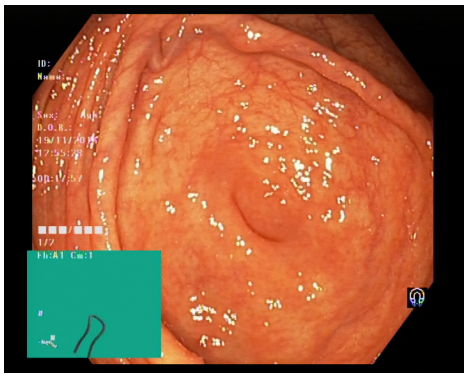
[1] Anemia is a decrease in the total amount of red blood cells



(a) Image of a healthy z-line where one can clearly see the transition from the pink colored mucosa of the eshophagus to the more red shaded gastric mucosa.



(b) Image of a healthy pylorus connecting the stocmach to the duodenum.

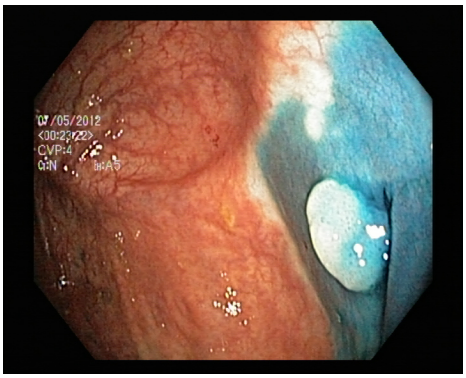


(c) Image of a healthy cecum located at the beginning of the large insteinte.

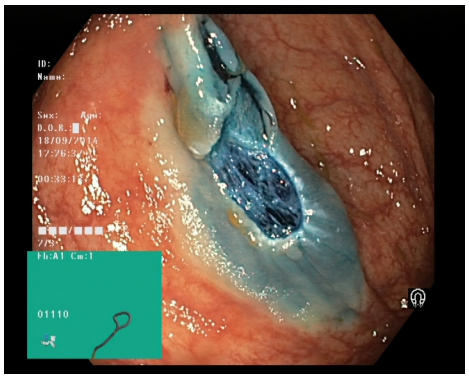
Figure 2.5: Sample images from each of the three classes of anatomical landmarks as part of (and taken from) the Kvasis (v2) dataset.

2.1.5.1 Z-line

The z-line is a section of the gastroesophageal junction (GEJ) which lies in the intersection between the esophagus and stomach. Endoscopically, it is the line formed by the transition from the white mucosa of the esophagus to the red gastric mucosa of the stomach. The z-line is an important landmark as it signals the exit of the esophagus, but it is also used as an area to detect disease. Specifically, the z-line is typically inspected in the diagnosis of GERD, which is caused gastric acid through the GEJ and up the esophagus. Looking at Figure 2.5a, we see the clear separation of the white and red mucosa forming the z-line.



(a) An image of a dyed lifted polyp marked for future removal.



(b) An image of a dyed resection-marign, which shows the presence of a previously removed polyp.

Figure 2.6: Sample images from each of the three classes of polyp removal markings as part of (and taken from) the Kvasir (v2) dataset.

2.1.5.2 Pylorus

The pylorus (Latin for "gatekeeper") connects the stomach to the duodenum, which is the first section of the small intestine. Using circumferential muscles, the pylorus regulates the number of intestinal contents passing through to the small intestine. Looking at Figure 2.5b we see the pylorus viewed from the perspective of the stomach, this is indicated by the pinkish color of the mucosa. The small hole contracts and retracts to regulate food passing into the duodenum.

2.1.5.3 Cecum

The cecum is a tube-like structure receiving undigested food from the small intestine and is considered the first part of the large intestine (colon). Reaching the cecum is the signals a complete colonoscopy, which is why recognition and documentation of the cecum is important. Figure 2.5c shows an image of a healthy cecum.

2.1.6 Polyp Removal Markings

As discussed in Section 2.1.4.3, polyps found in the lower intestine may be precursors to CRC and therefore removed even though they pose no immediate threat. A common technique of polyp removal is called endoscopic mucosal resection (EMR), which consists of injecting a lifting agent into the area surrounding the polyp, raising the polyp from the underlying mucosa, which is then removed using a snare. Detection and documentation of dyed polyps and resection margins are important to create complete endoscopic reports. Examples of these markings can be seen in Figure 2.6.

2.1.6.1 Dyed and Lifted Polyps

A Dyed lifted polyp is a polyp injected with a lifting agent to create a clear separation from the mucosa, making it easier to remove safely. To properly outline the injection site, blue dye is often added to the lifting agent. This is shown in Figure 2.6a, where we can see the blue dye surrounding the polyp. Various agents are used to create lifts, mostly depending on the size of the polyp. For small polyps (1 cm), saline is sufficient. For larger polyps, a more viscous agent is preferable, such as indigo carmine or methylene blue [35, 106].

2.1.6.2 Dyed Resection Margins

Dyed resection margins are the aftermath of a dyed-lifted polyp and are important to evaluate whether or not the polyp is completely removed. Residual polyp tissue may lead to continued growth and in worst case become malignant. Figure 2.6b shows an example of the dyed resection-site after polyp removal.

2.1.7 Quality of Colonoscopy Reporting

Documentation and reporting of colonoscopies play a pivotal role in the communication between healthcare providers and patients. Additionally, these reports provide a good source of data for use in research, quality assessment, and resource management. Despite the importance of these reports, and decades of work, studies find that they are often inconsistent, incomplete and lack standardization [51, 61, 83]. As mentioned in Section 2.1.2, colonoscopies are largely dependent on the endoscopists ability to detect signs of CRC in the form of its precursors (polyps). In severe cases, this is maybe a more important predictor than the key risk factors often associated with CRC (age, gender, etc.) [11]. This is supported by an analysis done on Canadian colonoscopy data, where there was a clear link between the quality measure and the endoscopists ability to detect CRC [5, 79]. Thus, a standardization and clear documentation practices could result in a lower rate of PCCRC. This is also supported by the European Society of Gastrointestinal Endoscopy (ESGE), which lists the standardization of the medical reporting in endoscopic procedures as a requirement [8].

In an attempt to measure and improve the quality of colonoscopy reports, the Quality Assurance Task Group of the national colorectal cancer roundtable (NCCRT) developed a standardized reporting and data collection system called CO-RADS [64]. This standard was created by compiling different colonoscopy reports from different hospitals, pulling the best features from each to come up with a single standard. Standardized systems have numerous advantages over non-standardized systems, including better communication of test results, standardization of terms and measure-

ment criteria, and the establishment of data systems that can be used for medical audits and continuous quality improvement (CQI). In CO-RADS, they define 25 key data quality indicators for colonoscopy reports and are defined as follows:

Patient Demographics and History

i Age

ii Sex

iii Other: Anticoagulation, antibiotic prophylaxis required, implantable defibrillator, or pacemaker present

Assessment of Patient Risk and Comorbidity

i ASA classification

Procedure Indication(s)

i Date of last colonoscopy

ii Previous most advanced histological lesion

iii Family history of CRC, adenoma, or inherited syndrome

iv Reason for examination

Procedure: Technical Description

i Date and time

ii Sedation with medication names and dosages

iii Extent of examination

iv Duration

v Documentation of cecal landmarks

vi Retroflexion

vii Bowel preparation (type and quality)

Colonoscopic Findings

i Mass/polyp (location, size, morphology, and method of removal or biopsy)

ii Other abnormalities

Assessment

i Based on history and colonoscopy findings

Interventions/Unplanned Events

i Type of event ± intervention

Follow-Up Plan

i Immediate follow-up and discharge plan (further tests, referrals, changes in medications, and follow-up appointments)

ii Recommendation for follow-up colonoscopy and tests

Pathology

Despite their efforts, the adoption of standardized electronic medical records (EMRs) and databases remain poor. In 2016, a study was conducted by Sharma *et al.* to review the current state of colonoscopy documentation [94]. The team collected a list of 30 papers referencing the quality of colonoscopy reporting and identified five themes for quality improvement. The five themes are as follows:

1. The need for standardized data models and templates.
2. The need for endoscopists to understand the value of complete and accurate documentation for effective clinical communication.
3. The need for standardized terminology.
4. The need for endoscopist performance feedback.
5. The need for appropriate health system use of data.

In the following sections, we discuss a few of these issues in more detail.

2.1.7.1 Standardization of Data Models and Templates

The quality of accompanying documentation often measures the mark of a quality colonoscopy. As this might be the only record of the performed procedure, it is critical that these reports are as complete and transparent as possible. Despite the clear guidelines and standards introduced over the last few decades (such as CO-RADS, MST, etc.), a large number of reports are still incomplete, often missing key elements from the procedure. A study done on more than 100 academic endoscopy centers in the U.S. revealed that key elements such as preparation quality and diagnostic interpretation were missing from roughly 40% and 58% of reports respectively [83]. One source attributed this problem to a lack of knowledge regarding the standardized reporting guidelines, and general poor agreement among community health workers [73].

A possible solution to this problem is the use of electronic reporting systems, where multiple health services have reported success in improving the quality of documentation through the implementation of such software [6, 47]. Palmer *et al.* also stated that they saw an increase in documentation quality among clinics/hospitals using automated reporting systems, but doubted that it would solve all of the standardization problems.

2.1.7.2 Understanding the Value of Documentation

As briefly mentioned in Section 2.1.7.1, poor quality colonoscopy reporting is partly due to disagreements and lack of knowledge surrounding the proposed guidelines and standards. This indicates that there is a general lack of training among endoscopists in the field of medical documentation, and a lack of understanding of how important quality reporting is when it comes to communication between patients, health care providers, and doctors. Although our assessment of endoscopic reporting so far has been in poor light, it is important to note that overall documentation of these procedures has steadily, but slowly, improved over the last 30 years. Starting in 1991, Mai *et al.* reviewed 1408 endoscopy and colonoscopy for deficiencies and found that only 28.7% included a follow-up plan [68]. Note that this was before the introduction of CO-RADS, but after the guidelines proposed by American Society for Gastrointestinal Endoscopy (ASGE). Similarly, in 2002 Robertson *et al.* found that approximatel£y 59% of colonoscopy in research-affiliated facilities included a procedure interpretation and plan. Lastly, a study done at the Mayo Clinic found that 81% of colonoscopy reports included follow-up recommendations and screening intervals [14].

2.1.7.3 Standardization of Terminology

In 1994, the ESGE, ASGE and Japanese Society for Gastrointestinal Endoscopy (JSGE) introduced the MST, which was a list "minimal" terms and descriptors that should be used to denote anatomical structures, endoscopic findings and their attributes, reasons of endoscopy, endoscopic diagnosis, procedures and adverse events. The goal of this was to establish a common vocabulary and structure for EEMR systems [20, 55]. Despite these guidelines, there is still a disparity between reports, even within geographically close clinics. Even within the state of Maryland, where Li *et al.* conducted a quality assessment on colonoscopy reporting, they found variations in descriptors such as some endoscopists classifying a 10mm polyp as large, and others classifying it as small.

2.1.7.4 Current Software Solutions

Software solutions for endoscopic reporting have been around since the 1980s in the form of simple computerized report generation but have since evolved into full electronic medical record databases incorporating comprehensive electronic practice management (EPM) software. Many of these systems include tools for collection of data through video and image capture directly from endoscopic procedures, going much further than just supplying reporting services. The implementation of these systems is essential as they make it easier to follow the MST and allow for quick analysis through searchable databases for clinical research and quality

Software	Company	Location	Discontinued
CORI	Clinical Outcomes Research Initiative	Portland, Oregon, USA	
EndoSoft	EndoSoft	Schenectady, New York, USA	
EndoPro iQ	Pentax Medical	Montvale, New Jersey, USA	
EndoProse	Summit Imaging	Lee's Summit, Missouri, USA	
EndoWorks	Olympus America	Center Valley, Pennsylvania, USA	2015
gMed	gMed	Weston, Florida, USA	
MD-Reports	Infinite Software Solutions	Staten Island, New York, USA	
ProVation MD	ProVation Medical	Minneapolis, Minnestoa, USA	
eMerge Endo	eMerge Health Solutions	Cincinnati, Ohio, USA	

Table 2.1: A brief overview of some of the most popular endoscopic electronic medical record systems.

improvement purposes. Table 2.1 shows a short list of some of the most prevalent GI reporting systems available. Even though the scope of this thesis is limited when it comes to the production of a complete endoscopic analysis and reporting system, it was still important to see what the current standards for such systems to evaluate the purpose of our included features better. As we did not have direct access to any of these systems described above, we will have to shuffle a full evaluation of these systems off to future work.

2.2 Machine Learning for Disease Detection and Diagnosis

Since 2012, machine learning has grown exponentially in its popularity and shown to produce state-of-the-art performance on various tasks including object recognition, language translation, and robotics. The current application of these algorithms can be found across a wide variety of different domains, including the field of medicine, wherein 2017, an Inception (v3) based CNN was able to diagnose skin cancer at the level of a trained dermatologist [27]. The success of these findings motivate the efforts of applying deep learning to other areas of medicine as well, with the purpose of aiding medical doctors in diagnosing different types of diseases. As for the scope of this thesis, we focus on the application of deep learning methods to detection and diagnosis of disease found in the GI tract.

In spite of the impressive results of deep learning, there are some challenges which make it difficult to implement in specific areas, especially fields where its output will be used as a basis for serious decision making. Firstly, deep neural networks generally need massive amounts of training data to perform well. This means we must have access to sizable datasets

of labeled data. This is one of the most significant hurdles when it comes to applying deep learning methods to medical image analysis, as there is not enough annotated data to train and evaluate a robust system for many use cases. There are multiple reasons for this; first of all, there is a general lack of medical experts dedicated to their respective fields, e.g., gastrologists, cardiologists, dermatologists, etc. This is a big problem when it comes to collecting annotated datasets as medical images need to be labeled and verified by experts within their field. Also, as with most medical data, there is a legal and ethical challenge of collecting and using other peoples data.

Secondly, although the general concept of deep neural networks is relatively easy to grasp, the internal processes and decision making of a network has become increasingly complex over the past few years, making it very difficult to interpret why a model produces a given result. This general lack of understanding has lead to neural networks being treated as a typical "black box"[2], where its users are only concerned with the data that is put in and the performance of its output. This may be acceptable when dealing with problems which have little to no consequence (such as classification of various cat breeds). But when it comes to diagnosing patients with life-altering diseases, where an incorrect diagnosis could be at the risk of a persons life, we must trust that the system can detect the objects in question and understand why it might make mistakes. Multiple methods have been proposed to open this "black box", some requiring a deep mathematical background in the theory of neural networks [72, 105], others focusing on easily interpretable visualizations [109, 112]. In this thesis, we focus on simple to understand interpretations through visualizations of various layers of a given network, making the production of given results easier to interpret by non-technical users such as medical doctors. In the upcoming sections, we provide a brief introduction to the field of machine learning, focusing specifically on two different types of neural networks, MLPs and CNNs.

2.2.1 Machine Learning

Machine learning is a sub-field of artificial intelligence (AI), which uses statistical techniques to give computer systems the ability to "learn" from data, without being explicitly programmed. A popular quote taken from Tom A. Mitchell's 1998 book, *Machine Learning* [70], summarizes the field of machine learning quite nicely;

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.* — Tom A. Mitchell [70]

[2] In the context of science, a black box is a device which can be viewed concerning its inputs and outputs, without any knowledge of its inner workings.

Machine learning can be applied in a variety of means, but to gain a good understanding of its most popular applications. Machine learning algorithms generally fall into three categories; Supervised learning, unsupervised learning and reinforcement learning. In the upcoming sections, we will give a brief introduction to each of these.

2.2.1.1 Supervised learning

Supervised learning is a collection of algorithms which learn from labeled data. This is done through an iterative process, where the algorithm predicts a given sample, then aptly shifts its internal weights based on how incorrect its prediction was. This process is continuously repeated until it either stops improving or reached a set boundary of iterations. One could argue that supervised learning algorithms such as CNNs sparked the renewed interest in machine learning applications with great results of Krizhevsky's *et al. AlexNet*, which won the imageNet large scale visual recognition challenge (ILSVRC) challenge of 2012 [56]. Algorithms part of the supervised learning family include support vector machines (SVMs), traditional neural networks (MLP), CNNs and decision trees to name a few. Common applications for supervised learning are image classification, language translation and speech recognition.

2.2.1.2 Unsupervised learning

Unsupervised learning is a collection of algorithms which learn from unlabeled data. It does this by automatically detecting patterns in the provided data and performing some task. Traditionally, the most common unsupervised learning methods have been cluster analysis. These algorithms analyze unlabeled data for common patterns between data points, then automatically groups similar data into "clusters". Typical applications of this can be sorting a large dataset of unlabeled images or automatically grouping people with similar tastes in a social network. A few examples of these algorithms are k-Means clustering, hierarchical clustering, and self-organizing maps.

In recent years, however, the popularity of unsupervised learning methods has shifted to focus more on generative models. The goal of generative models is to generate something entirely new. A more formal description could be using a dataset to learn the true data distribution and generate new data points with slight variations. Popular algorithms in this area of research include autoencoders, generative adversarial networks, and latent dirichlet allocations (LDAs). Possible applications for these algorithms include the generation of new media (such as music, images, etc.), text generation, and even as an aid in the development of new medicinal drugs [63].

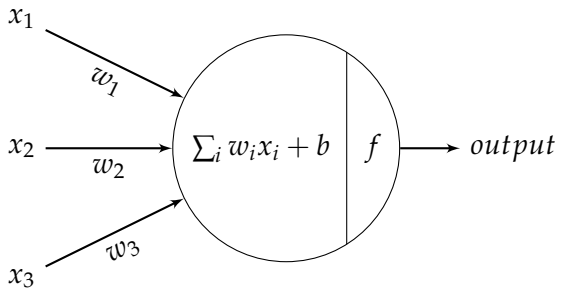


Figure 2.7: A visualization of a typical artificial neuron taking three inputs and producing a single output. Note that the vectors pointing into the neuron are weighted, and the weighted sum of the inputs are passed through an activation function before exiting the neuron.

2.2.1.3 Reinforcement learning

Reinforcement learning algorithms are greatly inspired by behaviorism, in that they consist of agents trying to maximize some reward in a given environment (i.e., an agent learns from the consequence of its actions). Popular algorithms within this family of machine learning are Q-learning, state-action-reward-state-action (SARSA) and Deep Q-networks [71, 87, 108].

These algorithms are often classed outside of the traditional supervised and unsupervised categories, as it has specific proprieties which make it suitable for neither. It does not fit into the class of supervised learning as it does not strictly learn from labeled data, but from a response based on an action taken. Neither is it unsupervised as we already know a target reward for which the algorithm should optimize.

2.2.1.4 Deep Learning

Deep learning is a broader class of algorithms which directly relate to artificial neural networks (ANNs). Deep learning is used to denote "deep" neural networks, meaning neural networks consisting of many internal layers. A common misconception is that deep learning only relates to supervised learning problems. This is not true as there are plenty of "deep" algorithms related to reinforcement and unsupervised problems as well, such as Deep Q-networks and Deep belief networks [48, 71]. As for the work produced in this theses, we will only be using deep learning in the context of supervised image classification problems, using "deep" CNNs.

2.2.2 Neural Networks (Multilayer Perceptrons)

ANNs or neural networks are computational models, loosely based on the neurological constructs that make up the "animal" brain. Historically, the development of networks has been strongly motivated by this biological

system but has since diverged and become more a principle of engineering with the goal of achieving excellent results in machine learning tasks. Similar to how humans learn, neural networks learn by example. However, unlike humans, neural networks generally need thousands, if not millions, of examples before being able to perform nearly as well (although there is an exciting field of research called one-shot learning, where networks only need a few examples [32]). This section will cover the basics of supervised neural networks, starting at the very basic building block, the neuron, and build up towards how these are organized in networks to perform the machine learning tasks for which they now are so well known.

2.2.2.1 Perceptron

Artificial neurons, or neurons, are the basic computational units of a neural network. Typically, neural networks may consist of tens of thousands, or even millions, of neurons, each working in tandem to solve one specific problem. To gain some intuition into why these networks work so well, it is important to have a basic understanding of how each neuron works individually. In Figure 2.7, we a typical representation of a generic neuron. It takes three weighted inputs denoted by the three vectors pointing to the neuron, note that there are no restrictions on the number of inputs. The neuron performs a weighted sum of its inputs, together with an additional bias term (typically -1 or 1). The role of the bias term is to shift the output in a negative or positive direction. The result of this weighted sum is then passed through a function (often called the activation function and is denoted by f in Figure 2.7), the purpose of this function is to add some non-linearity to the solution. This is arguably the most important piece of the neuron as, without it, a full network of neurons would work no better than a single one. This function is what differentiates neurons from each other, as there have been a variety of activation functions proposed over the years. After passing through the activation function, the output is either sent to a neuron in the proceeding layer or given as output of the network.

$$f = \begin{cases} 0 & \text{if } \sum_i w_i x_i + b \leq 0 \\ 1 & \text{if } \sum_i w_i x_i + b > 0 \end{cases} \tag{2.1}$$

To get a better understanding of how these neurons may be used in practice, we look back in history to the implementation of some of the first neural networks implemented, the Perceptron. Based on the work done by Warren McCulloch and Walter Pitts (therefore also sometimes referred to as McCulloch-Pitts Neuron), Frank Rosenblatt developed the Perceptron between the 1950s and 1960s [84]. The Perceptron is a binary linear classifier based on threshold logic, structured like the typical neuron shown in Figure 2.7. The activation function of the Perceptron is a simple threshold algorithm which fires if the sum of its inputs is greater than 0 (this computation is expressed in equation 2.1). At the time, Perceptrons

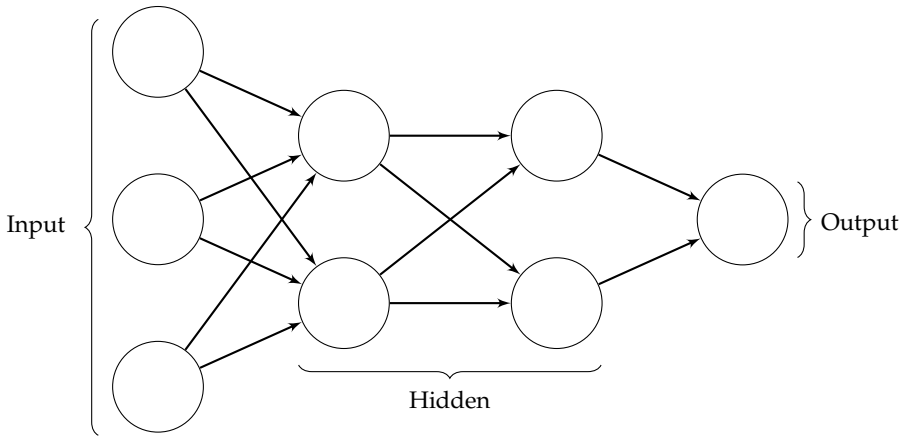


Figure 2.8: A three-layered multilayerer perceptron containing a total of five computational neurons (excluding the input).

showed much promise in being used as learning devices, from which Rosenblatt wrote a book on many different Perceptrons and the various applications. This excitement diminished in the late 1960s when Minsky and Papert published a book called "Perceptrons", which analyzed what Perceptrons could do and showed their limitations. The main issue with the Perceptron was that it was limited to only solving linearly separable problems, meaning that it can only distinguish between classes that can be separated by a single straight line. A typical example used to show this limitation is taught a perceptron to learn the bit-wise operations AND, OR, and XOR. A single Perceptron can quickly learn the AND and OR operation, but it is impossible to learn it the XOR function.

2.2.2.2 Multilayer Perceptron

The solution to the XOR problem lies in the fact that we can combine multiple Perceptrons in sequence to gain some non-linearity. By using a two-layered network of Perceptrons, two Perceptrons in the first layer and one the output layer, we are successfully able to combine the outputs of each node to solve the XOR problem. In fact, we only require three layers to build a neural network that can approximate any function according to the universal approximation theorem [19]. Arranging the Perceptrons in such a way we get what is called a MLP, and is what we today consider a "vanilla" neural network. Looking at Figure 2.8, we see each layer is categorized into one of three classes; input, hidden, and output. The input layer is where data passed into the network and forwarded to the first hidden layer (or output layer in the case of a single-layered network). There are no learned parameters or computations performed in this layer (and is therefore excluded when referencing the size of a network). The output layer is always the last layer of the network and is commonly domain-specific, meaning the number of neurons equal the number of classes to be classified. In the case of classification, the output layer is typically a

classifier (such as softmax), containing a node for each classification class. The hidden layers are what lies between the input and output layers. Note that data is always input into the input layer, passes through the hidden layers, and is output in the output layer. This puts the MLP together with the feed-forward class of neural networks. These networks employ a unidirectional data flow, meaning there are no cyclic connections between neurons. The alternative to this class of networks is recurrent neural networks, which allows networks to make previous decisions affect the output of a neuron (sort of like a memory of previous decision).

2.2.2.3 Training a Neural Network

In the previous section, we presented the typical neural network architecture in the form of a MLP. This covered how neural networks are structured as layers of neurons, each adding some non-linearity to the solution. But we are still missing a crucial part of what makes neural networks work so great, how a network learns through updating the weighted connections between each layer. This section will cover the algorithms used to update the weights of a typical neural network. This includes a look at backpropagation, a few optimization functions and how we calculate the loss of a network.

For supervised neural networks to learn, we need some way of calculating how wrong the predicted output of the network is. This is done in a variety of ways and is commonly referred to as the loss function of a network (also known as the error function and cost function). The most common method of calculating the loss for a modern neural network is through the use of an algorithm called cross-entropy, which measures the difference between the predicted output and the actual ground truth.

$$C(x, y) = -\sum_{i} x_i \log y_i \tag{2.2}$$

Looking at the Equation 2.2, we see how cross-entropy is calculated where x denotes the predicted output, y denotes the ground truth, and i denotes the class in question. This function has to attributes which makes it work particularly well for a loss function. Firstly, it is non-negative, meaning values will never go below zero. Secondly, the better the network performs, the closer to zero the loss will be (a loss of zero would be a perfect score), making it mainly fit for optimization using the typical optimization method of gradient descent.

Now that we have a way of calculating the score of our network, we need some way of minimizing this loss. The functions used for minimizing the loss is often referred to as the networks optimization function, of which the most commonly used algorithms are based on variations of gradient descent. Gradient descent can primarily be split into three variants; batch gradient descent, stochastic gradient descent (SGD) and mini-batch

gradient descent. The traditional gradient descent algorithm (batch gradient descent) takes the entire dataset into account when calculating the gradient of the loss function with regards to the weights w. This can be seen in Equation 2.3, where η denotes the learning rate (how much the weights should update), and $\nabla_w J(w)$ means the gradient of the loss function accounting for the entire training set. This method of gradient descent is typically not used in practice as it is particularly slow and is limited by hardware memory requirements needing to fit the entire training dataset. Additionally, batch gradient descent has the issue of converging to the nearest lowest valley.

$$w = w - \eta \cdot \nabla_w J(w) \tag{2.3}$$

SGD (Equation 2.4) solves the issues of slow updates and large memory requirements by performing updates for each training sample (each item in the dataset). By updating for each training sample, we are also able to fluctuate the results to potentially find a better local minimum than that which would be found by batch gradient descent. However, this fluctuation may cause some issues as well. By continually optimizing for the local minimum, SGD often overshoots the local minimum, making it difficult to find the exact best solution. This issue is partly solved by a technique known as momentum, where we gradually decrease the learning rate of the optimizer as the number of epochs grows.

$$w = w - \eta \cdot \nabla_w J(w; x^{(i)}; y^{(i)}) \tag{2.4}$$

Mini-batch gradient descent is a compromise between the two methods, updating weights based on a set number of data samples (a batch size). This is shown in Equation 2.5, where we that n denotes the of samples included in each weight update. This is the most popular variant of gradient decent and has shown to produce the best results.

$$w = w - \eta \cdot \nabla_w J(w; x^{(i:i+n)}; y^{(i:i+n)}) \tag{2.5}$$

The above-explained variants of gradient descent are rarely used as is. In most cases, we use extended algorithms which help us help us in making our models converge. An extended look at there algorithms is outside the scope of this thesis, but to name a few popular ones we have Adagrad, Adam, and Nadam [22, 23, 54]. For the most part, these extended algorithm helps us tune the learning rate to have a higher guarantee of convergence (such as lowering the learning rate after some iterations).

Now that we have explained how to calculate the score of a network (loss function), and how to update a given set of weights based on its gradients (optimization function), we now look at the method of

applying these methods to our network through backpropagation [86]. The backpropagation algorithm propagates the error (loss) calculated by the loss function back through the network to optimize the weights to be closer to the desired output.

2.2.3 Convolutional Neural Networks

Similar to the MLPs described in the previous section, CNNs are feed-forward neural networks which learn through similar means (i.e., updating weights through backpropagation). What sets a CNN apart from traditional neural networks is that it uses a grid-like topology, making it especially adept at processing data of multiple dimensions such as samples taken at regular time intervals or the pixel dimensions of an image. Although a CNN has a variety of use cases, it is today most well known for its performance on the field of image classification. In the past few years, CNNs have continuously achieved excellent results in many areas of computer vision, including its domination of the ILSVRC since 2012. The secret behind the CNNs success lies in its ability to express computationally large models while keeping the number of parameters[3] relatively low. This is all done by introducing two new concepts to the traditional network architecture, convolutional and pooling operations. But before diving into the details of how these two operations work, we first give some background into why we typically do not want to use a standard neural network for image classification.

Recall a MLP consists of one input layer, an optional number of hidden layers, and one output layer. The input layer takes a single vector as input and transforms it through a series of hidden layers before it is classified in the output layer. Each hidden layer consists of a set of neurons, each fully connected to every neuron in the previous layer. Let us first calculate the number of parameters used by such a network when training on a dataset consisting relatively small images, such as CIFAR-10 where images are only of size $32 \times 32 \times 3$ (32 wide, 32 high, 3 color channels). The input would be in the form of a single vector with the size $32 \times 32 \times 3 = 3072$, meaning with just a single hidden layer consisting of one neuron we already have 3072 learnable weights. If we were to use a network similar to the one in Figure 2.8, the total number of learnable weights would be equal to $(32 \times 32 \times 3 \times 2) + (2 \times 2) + 2 = 6150$. This might not sound like much, but we typically do not want to train on images of such small size, and we typically want more than two neurons in a single hidden layer.

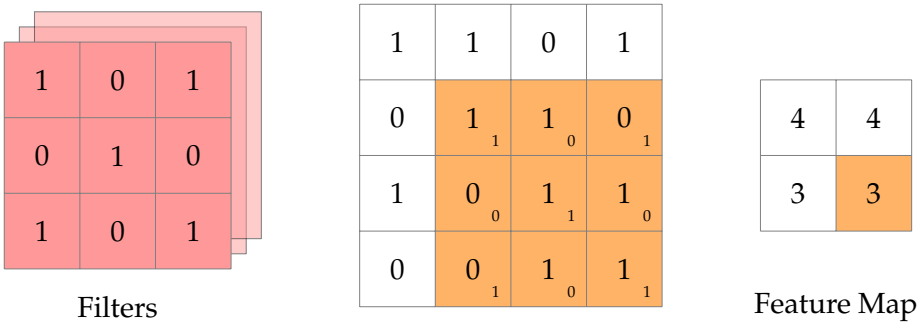


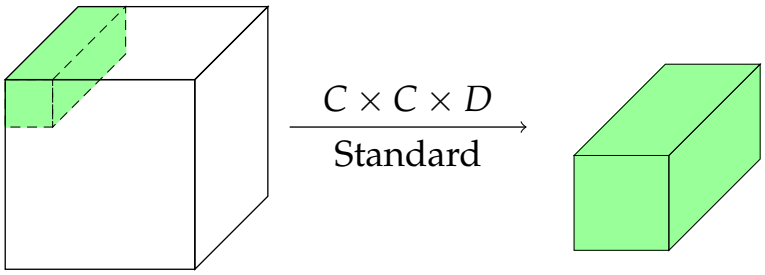
Figure 2.9: This diagram shows an example of a convolutional operation on a $4 \times 4 \times 1$ image using a kernel size of $3 \times 3 \times 1$ and a stride of 1.

2.2.3.1 Convolutional Layers

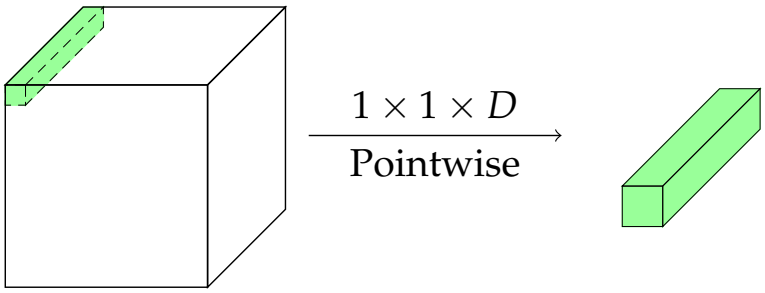
The convolutional layer is arguably the most important piece of a CNN. It is centered around the convolution operation, which sort of blends two functions together to produce a third function (which in the context of CNNs is a mixture between the input function and the kernel function). Its parameters are a set of learnable filters (Sometimes called kernels) with a set spatial size (i.e., width and height), typically spanning the entire depth dimension of its input. During the forward pass, each filter slides across the input, performing a dot product between the weighted filter and input at any given position. The result is what is often referred to as a feature map. This is illustrated in Figure 2.9 where we see a filter of size $3 \times 3 \times 1$ slide across a $4 \times 4 \times 1$ input to produce a 2×2 feature map.

2.2.3.2 Depthwise Separable Convolution

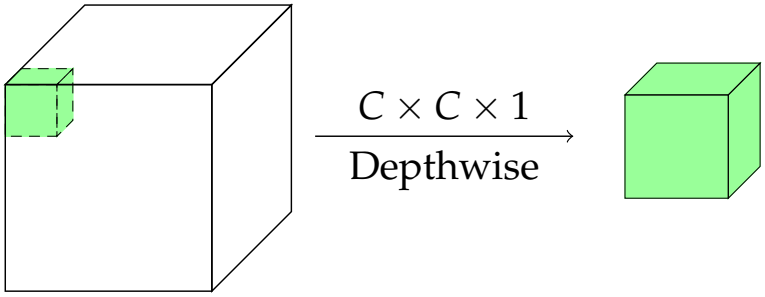
In recent years, a new type of convolutional operations has become popular among CNN architectures, notably Inception and Xception. This new form of convolution is called *depthwise separable convolution* and splits separates the standard convolution into two separate steps. Recall that the standard convolution convolves simultaneously across both spatial and cross-channel dimensions. In depthwise separable convolution, these two operations are separated, by first performing a depthwise convolution, then following it up with a pointwise convolution. Figure 2.10 shows a visual depiction of how this works in three-dimensional space.



(a) An illustration of a standard convolutional operation.



(b) An illustration of a pointwise convolution.



(c) An illustration of a depthwise convolution.

Figure 2.10: A set of illustrations demonstrating three types of convolutional operations. Note that pointwise and depthwise convolution is used extensivley in the Xception CNN architecture

2.2.3.3 Pooling Layers

Pooling layers, sometimes referred to as downsampling layers, are commonly placed in-between convolutional blocks [4]. Their purpose is to reduce the spatial size of the internal representation, thus reducing the number of parameters, number of computations and the ease of large neural networks. This is typically done through either a max or average pooling [114], which similar to the convolution operation works as a sliding window across the representation. The values within the window are

[3]The parameters of a neural network is typically a reference to the parameters learned by the model itself, this is different from the hyper-parameters which are chosen by the networks implementer.

[4]Convolutional blocks are a shorthand way of describing blocks containing many successive convolutional layers

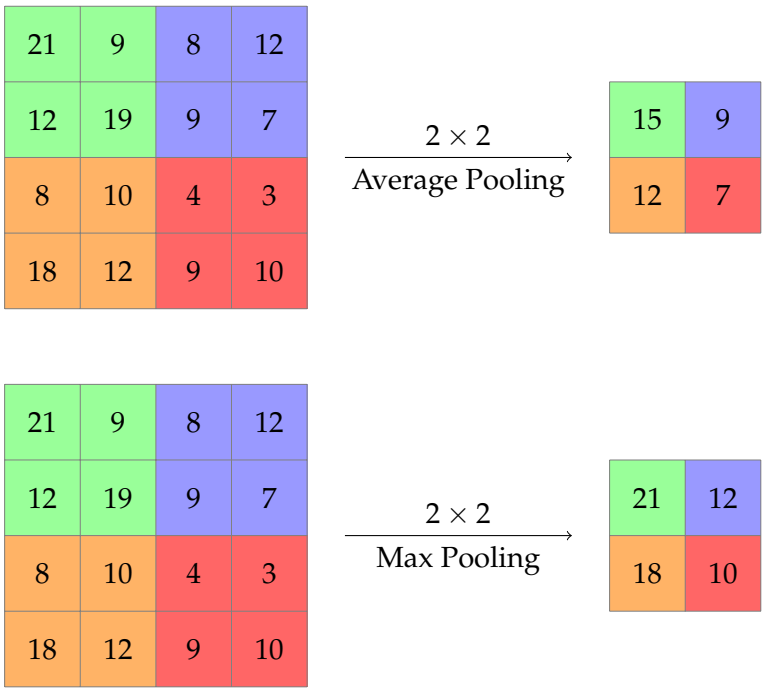


Figure 2.11: A visual representation of how the max and average pooling operations work on a two dimensional matrix using stride and pool size of 2.

"pooled" using either operation and placed in the corresponding space of the output. The size and stride of the window is set during the implementation of the network and is most commonly set to 2 and 2 respectively. This is shown in Figure 2.11 where we see an example of the two most common pooling operations, max and average pooling, sliding over the input with a spatial size of 2 and stride of 2. Note that there are no weights tied to the polling layer, meaning its only purpose during backpropagation is correctly routing the gradients.

2.2.4 Deep Learning in the Medical Field

With the ongoing success of applying deep learning methods to almost any field, it is only natural to start using it to mission-critical areas as well. In the case of medicine, deep learning has already found its footing in an array of different medical tasks such as disease detection, drug discovery, radiology, and epidemic outbreak prediction [3, 10, 49, 69]. This is just a few examples, but researchers believe that deep learning has much promise within the field of medicine [39]. As we have previously discussed, our work will be focused on applying deep learning methods to the area of GI disease detection and diagnosis.

Applying deep learning to the GI tract is relatively new, with most research focusing on the detection and documentation of polyps [82, 107]. This is understandable as polyps pose the highest risk for CRC.

Additionally, polyps are already well documented beforehand, meaning there is already extensive datasets containing annotated polyp images. This is not to say that it is the only area of deep learning application, research also shows that deep neural networks may be used to detect lesions, detect bleeding, and detect various other diseases found in the GI tract [50, 77, 115]. In the upcoming two sections, we discuss two issues with applying deep learning methods to the fields of medicine.

2.2.4.1 Issue of Interpretability

A common issue of applying deep neural networks to the field of medicine, among other mission-critical areas, is that they are neither easy to understand or easy to interpret. We do not necessarily mean that their outputs are difficult to understand, but the process that produces this output is generally referred to as a "black box", especially among non-technical users. This lack of understanding can often lead to a trade-off between intelligibility and accuracy. Where traditional statistical models such as logistic regression and decision trees are preferred over the high accuracy yielding neural networks.

A good example of this took place in the 1990s, where a large multi-institutional project was funded by Cost-Effective HealthCare (CEHC) to evaluate the use of machine learning algorithms on important problems within health care, using the prediction of pneumonia risk as an example [9, 17]. The goal of this study was to predict the probability of death for patients with pneumonia such that high-risk patients could be submitted to the hospital, while low-risk patients were treated locally before being sent home. After conducting their experiments, they found that the neural network based algorithms outperformed the traditional methods such as logistic regression by quite a large margin (0.86 compared to 0.77). Although the neural network based model was undoubtedly the most accurate one, the team decided it was too risky to apply this method to real-world patients, and adopted the logistic regression method instead because the rules for classification were more intelligible than those for the neural network. This goes to show that having a good understanding of the internal workings of applied algorithms is an essential piece in gaining the trust and adoption of the medical community. Now, even though this study was conducted in the mid-1990s, deep learning based algorithms have not become any more interpretable, one could argue that they have become less so, as new architectures using hundreds of layers have become commonplace in the high performing models.

2.2.4.2 Issue of Data

As we briefly discussed in Section 2.2.2.3, deep neural networks typically need to train on large amounts of data to obtain a high level of classification performance and generalizability. This is among the most significant

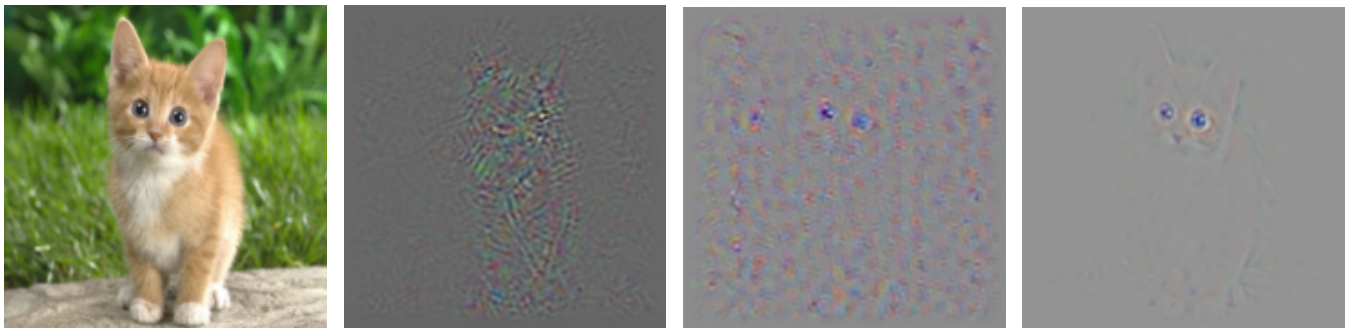
hurdles when it comes to its application in the medical field. Finding large public datasets related to medical imaging is difficult in itself, but this dataset must also refer to the problem at hand, i.e., it may be easier to find a large dataset of polyp related images (as they are commonly documented) than a dataset containing images of inked polyps (as they are not commonly recorded). This is an issue as both datasets would be meaningful in their own sense, but cannot be used interchangeably. In addition to this, there is the issue of having the data annotated by a trained expert within each respective field, i.e., preferably a trained gastroenterologist for GI related images. This is mainly due to a lack of experts within the respective medical field willing to perform the tedious work of manually annotating thousands of medical images. Work has been done to make this process easier, such as batch annotation through the use of various unsupervised clustering methods. But there is still an issue with fundamental ethical and privacy concern when handling any form of medical data.

2.2.5 Opening the Black Box of Neural Networks

As previously mentioned, deep neural networks have become a crucial tool in a variety of applications and extensively used in a wide range of academic research. However, it is often essential to verify that for a given task, the measurement accuracy and performance is due to the specific problem at hand, and not due to artifacts or noise present in the training data. This has been a frequent criticism of deep learning methods as they are often used a "black box", without question of whether or not the predicted classes are in fact being detected. For some areas, this might be acceptable. But in mission-critical fields such as medicine, the criminal justice system, and financial markets, this has been problematic as being able to understand, validate, and trust the output of these models is very important. The consequence of this is that many critical areas choose to implement simpler, less accurate models, as they are often more intelligible when compared to most deep learning methods. With the performance of deep neural networks often tied to their "deepness", we do not foresee this problem being solved by its own. With a goal of gaining a deeper understanding of the decisions made by a deep neural network, motivated by achieving the right amount of trust and increased intuition on ways to boost the performance of these methods, a new field of research has emerged with a focus on the interpretation of deep neural networks.

2.2.6 Visualization Techniques

To build trust among those who use the systems built on neural networks, we must develop transparent models which can explain why they produce a given result. These explanations must be easy to understand as most users will neither have the technical or mathematical knowledge



(a) Input Image (b) Backpropagation (c) Deconvolution (d) Guided Backpropagation

Figure 2.12: A comparison of three gradient based saliency maps. Images taken from Selvaraju's blog [91] [5]

to understand how such as system works at a fundamental level. A commonly used practice among deep learning practitioners is using visualizations such as heat maps and saliency maps to gain a better understanding of a models faults, thus increasing the intuition to rectify these issues through additional training. These visualizations, however, are often lost when serving the model to the non-technical end users, i.e., visualizations are only used in the production of high-quality models, not used to give end-users a better understanding of why a network returns a given result.

But before we start looking at various visualizations, we must first define what makes a good visualization. As CNNs are classification models, visualizations which are class specific are preferable, i.e., visualizations which can be used to localize a target class within the given image. Additionally, these visualizations should capture the minute detail of the image, i.e., visualizations should be high-resolution without losing quality. Various methods have been proposed over the past few years, but as for the scope of this thesis we have decided focus solely on the visualization of deep CNNs in the context of image classification, although much exciting research is being done on understanding other types of networks as well [52]. In the following few sections, we give a brief introduction to various types of visualization methods in the form of saliency maps and heat maps.

2.2.6.1 Generating Pixel Level Saliency Maps

A standard issue among early CNN visualization methods was that they were mostly limited to the initial layers where the projections can be mapped directly back to the input image pixel space. Interpreting latter layers are tricky, as typically represent features which are more

[5] https://ramprs.github.io/2017/01/21/Grad-CAM-Making-Off-the-Shelf-Deep-Models-Transparent-through-Visual-Explanations.html

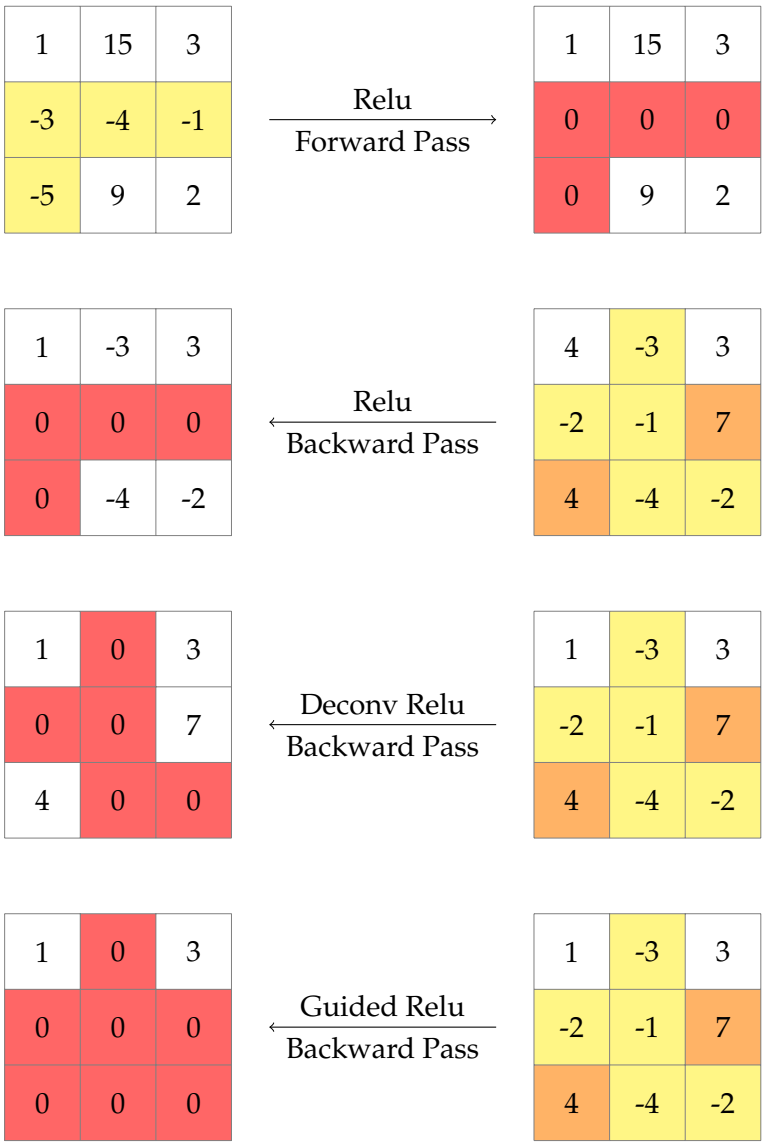


Figure 2.13: This is a diagram showing how the guided ReLU operation works compared to a traditional ReLU operation. Note that in the guided ReLU operation, all values under 0 are set to 0 (including the values under 0 during the forward pass). Note that the yellow colored cells represent negative values, orange colored cells represent the thresholded valuies in the forward pass, and red colored cells represent the values set to 0 in each operation.

complicated and correspond to a combination of lower level features [25]. In 2013, Zeiler *et al.* proposed a method which aims to resolve this issue by presenting a way of mapping these activations back to the input pixel space, thus creating feature maps directly correlated to the input image [110]. This mapping is done using a deconvolutional neural network [111], which in simple terms a can be described as a reversed CNN, i.e., instead of mapping pixels to features, it does the

opposite. The deconvolutional neural networks were initially proposed as a form of unsupervised learning but are here merely used as a tool to inspect an already trained CNN. Without getting too specific on the exact operations, the deconvolutional neural network employs deconvolutional layers and depolling layers in place of convolutional layers and polling layers respectively.

As we discussed in Section 2.2.3, a typical CNN consists of convolutional layers, followed by some non-linearity (often rectified linear unit (ReLU)), and is then optionally followed by a polling operation. For deconvolutional based visualizations, we attach the deconvolutional counterpart to each of the corresponding convolutional layers, i.e., we attach deconvolutional layers to convolutional layers and depolling layers to polling layers. To generate the visualizations, one selects a single feature map from a selected convolutional layer, then pass this filter through the deconvolutional network. The result is a generated image showing what is detected in a given feature map. As we previously discussed, the initial layers of a typical CNN detect basic features such as edges and colors, with features becoming more abstract as layers progress. This theory can be visualized using the deconvolution approach using feature maps from the first layers and the last layers. Looking at Figure 2.12c, we see some example visualizations using the deconvolutional approach. This is an improvement over the straight mapping of activations, but do not exhibit any class-specific properties.

In the same year Zeiler and Fergus [110] proposed the deconvolutional method of generating image feature map visualizations, Simonyan, Vedaldi and Zisserman [96] proposed a generalized method displaying class-specific properties together with a much simpler implementation. This generalized method is based on the gradients produced during backpropagation by taking the derivative of the projected output of a CNN with respect to an image. This greatly simplifies the method proposed by Zeiler et al. as we only need a single backward pass to produce the saliency map. Similar to the deconvolution approach, this gradient-based method aims to show which pixels are most significant in the classification of the given image using a certain class. However, as we see in the example image shown in Figure 2.12b, mapping "raw" gradients produce images that are quite vague, noisy and not distinct. Springenberg et al. [98] proposed a solution to this problem. By making a slight change to the ReLU activation, they were able to reconstruct images which were significantly more accurate, especially for the last layers of the network. An example of these improved gradients can be seen in Figure 2.12d.

In sum, the main difference between the three methods of generating saliency maps is the way they handle the non-linearity, i.e., ReLU. Figure 2.13 shows an overview of the three variations of ReLU used. Of the three methods discussed, we use the guided backpropagation method as this yields the best results.

2.2.6.2 Generating Class Discriminate Activation Maps

A popular computer vision task is the localization of objects through bounding boxes or object segmentation. Typically, in order to train a network to create these boxes, we need the standard annotated image dataset in addition to a dataset containing the localization of each object, i.e., a corresponding dataset showing where the class in question is located inside the given image. Under training, the loss is then calculated by how well the predicted bounding box compares against the ground truth. As some fields already have an issue with finding datasets containing labeled data, this additional requirement is even more cumbersome. In 2015, Zhou *et al.* discovered that the layers of a CNN could act as object detectors without any supervision, i.e., without the need for annotated bounding boxes [113]. However, this ability was lost when using fully-connected layers before classification. A recent approach in reducing the parameters of a CNN is replacing fully-connected layers with polling layers. It turns out that using global average pooling (GAP) layers in place of the fully-connected layers before classification retains the object localization features. These visualizations were coined class activation maps (CAMs) because of their class discriminate features.

Building on the work done by Zhou *et al.*, Selvaraju *et al.* proposed a new method of generating these CAMs without the need for altering the existing architecture [92] called gradient-weighted class activation mapping (grad-CAM). By performing GAP after calculating the gradients of a given convolutional layer, Selvaraju *et al.* was able to apply the CAM technique to a variety of different architectures and for entirely different purposes. In essence, to produce these visualizations we first calculate the gradient of target class C with respect to the feature maps of a convolutional layer. These gradients are then run through a GAP operation to get the most important weights of class C. The resulting weights are then passed through a ReLU function to produce the final grad-CAM. It is important to note that if the existing architecture is already CAM compliant, grad-CAM produces the same localization maps as a standard CAM. In the same paper, Selvaraju *et al.* proposed another visualization method of combining the class discriminate properties of the grad-CAM with the pixel level quality of saliency maps. To do this, one performs a linear combination of the grad-CAM with saliency map, specifically guided backpropagation, to produce what they called a guided grad-CAM. For the work done in this thesis, we use both the grad-CAM and guided grad-CAM to give users a better understanding of the analysis performed by our trained neural network based models.

2.3 Summary

In this chapter, we discussed the background and related works of the fields related to our three primary research objectives. This included a

case study on the GI tract, where we looked at various types of endoscopy, two of which (colonoscopy and gastroscopy), are the current conventional methods of inspecting the upper and lower GI tract. Regular screening of the GI tract is essential for the discovering of disease, which in some cases may be life-threatening.

In conclusion, we learned that the current state of endoscopy reporting is generally considered weak, and depending on the complexity of the report (number of notable findings), they can take upwards towards 15 minutes or more to make. These aspects lend themselves well for automatic generation, as much of this work is repetitive, it can be much improved through the use of modern automation methods such as deep learning. However, for such systems to be trusted and accepted into the medical domain, the underlying analysis must be understandable and interpretable by the medical experts using them. Based on these open questions, we researched and developed an automated reporting system which makes the neural network based analysis transparent through the use of various intermediate visualizations, and enables the user to use this information to generate an editable, standard-compliant medical examination report. This system, called Mimir, is presented in the next chapter.

Chapter 3

Mimir: An Automatic Reporting System for Endoscopic Examinations

With the goal of aiding medical doctors in the analysis and documentation of GI endoscopies, and to further increase the understanding and trust in a neural network based automated detection systems. We developed a system, *Mimir*, which attempts to make the analysis performed by a deep neural network understandable through intermediate visualizations of a CNNs inner layers [44–46]. As we discussed in Section 2.2.4, understanding the algorithms behind a diagnosis is essential because of the high-risk often associated with medical decisions, e.g., a cancer diagnosis. Also, Mimir includes tools for generating endoscopy reports through a web-based interface, with options for attachment of images related to the systems suggested diagnostic.

The motivation behind this system is broadening the acceptance of deep learning within the medical community, and improving on the general lack of standardization and quality among colonoscopy reports (which was discussed in Section 2.1.7), something which an automated system may very much improve. This relates back to the three research objectives stated in Section 1.2. A complete overview of Mimir can be seen in Figure 3.1, which shows the expected workflow, beginning at the endoscopic procedure (colonoscopy, gastroscopy, etc.), and ending at the generated report. Please note that the system is not complete, it is merely a prototype and requires further work to be production ready in a medical environment. As such, we will be discussing Mimir as it is in its current state. As of now, Mimir mainly consists of three main functionalities, each related to the requirements set in Section 1.2.

1. The system was designed to aid medical doctors in making informed decisions regarding the diagnosis of diseases found during examin-

ations, such as diagnosis of disease found in the GI tract during a colonoscopy.

2. Mimir creates automatic reports based on the automatic analysis of images and videos and reduces the time spent on the administrative tasks that follow an endoscopic examination, e.g., documentation by written reports. This is shown in Figure 3.1 where a doctor uses the system to understand the analysis done by the neural network and use this information to reach a diagnosis and generate the accompanying report.

3. Mimir can be used by researchers and engineers designing deep learning architectures such as CNNs to gain a better understanding of the evaluation and reactions of their models, e.g., by understanding which parts of an image confuse the algorithm and if additional pre-processing steps are needed.

This chapter will give a detailed look of Mimir, starting with a technical overview, we discuss the various tools and technologies used for the implementation of Mimir's server and client. Here we will mostly present and argue the use of these technologies, showing how they affected the development of Mimir, and on what premise they were chosen. With a basic understanding of the technology behind Mimir, we present a detailed look at each tool included in the system. Starting with the neural network dissection tool, we show how it may be used to gain a deeper understanding of a CNNs decision process through analysis of the intermediate layers of a CNN. Secondly, we look at the report generation tool, which aims to assist medical doctors in writing endoscopy reports through a what you see is what you get (WYSIWYG) interface. With a thorough look at each tool complete, we end this chapter by looking at some use case scenarios which we imagine Mimir would be a good fit.

3.1 Mimir

Mimir [44–46] is structured around a client-server architecture. The benefit of this arrangement is offloading the computational costs of deep learning onto a reasonably powerful central server, which any client can make use of without having to worry about hardware requirements. Additionally, our client is implemented as a web application, which has the added advantage of being easily accessible from any device that supports a modern web browser. In our case, the client and server code are separated into their respective directory and workflow, making it easy to develop and run them independently.

As with most modern software projects, Mimir is not written from scratch, but with the aid of various tools, libraries and frameworks. To keep a certain level of quality and maintainability, we decided to set some ground rules to which technologies would be used in the development of

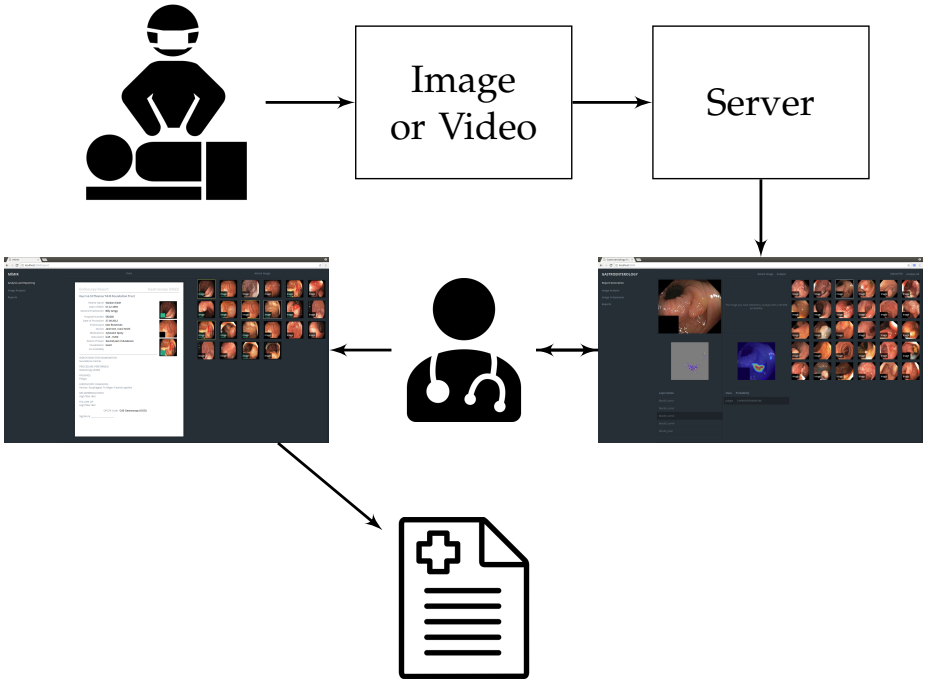


Figure 3.1: This diagram shows a complete overview of the Mimir system. Starting at the endoscopic procedure, image or video data is collected and sent to a central server. This server is accessed by a medical doctor through a web-interface where he/she can perform analysis on the endoscopic media. Based on the performed analysis, the doctor can generate a report using the WYSIWYG editor and produce the final endoscopy report.

Mimir. These criteria are not necessarily there to be followed slavishly, but more to make a mental note of whether or not a certain technology is necessary. The criteria are listed below:

- The technology in question should be mature, and widely used in the industry. The reason behind this is that we want to use tools which are thoroughly tested,x where we do not need to waste time on anything else but our bugs. In addition to this, it is important that the software is (to an extent) easily maintainable by developers other than the author. This is especially true for open-source software as it is available to the public.
- The technology in question should save us significant time in one way or another. If we can implement the functionality ourselves, we will do so. The main reasoning behind this is that by limiting the number of dependencies, we have more control over the software that we develop. In addition to this, we would like to avoid bloat that often comes with libraries where we only need one piece of functionality.

With these requirements in mind, we started the development of Mimir. In the upcoming sections, we take a closer look at the front-end and back-

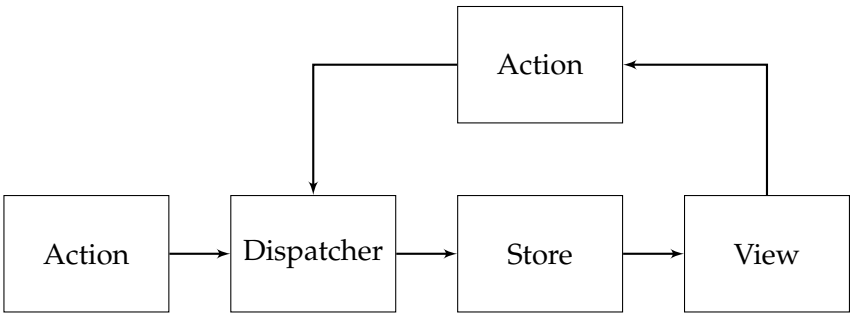


Figure 3.2: This diagram shows how data flows when structuring a project using the Flux pattern. We see that it starts with an action being triggered, this could be through some deliberate interaction done by the user or an automatic request sent by the application. This action is passed into a dispatcher, which signals the store to update the internal state according to the described action. As the store is updated, the view notices a change in its state and updates accordingly. Note that it is impossible for the view to directly modify the store, all steps must be performed in sequence.

end implementation of Mimir, with an additional part for the use of deep learning libraries.

3.1.1 Front-end Architecture, Tools and Technologies

Mimir's front-end (client) is developed using a combination of HTML5, Sass, and JavaScript. HTML5 is the fifth and latest version of the Hypertext Markup Language (HTML) standard, and is the standard markup language for creating web pages and web applications. Sass is a well known and widely used extension to the Cascading Style Sheet (CSS) language, which adds some additional features such as support for variables, functions and module imports. It is fully compatible with all versions of CSS, so the cost of including this in our project negligible. Mimir is written using single page application (SPA) principles, meaning that a single web page is loaded and dynamically updated as the end-user interacts with the application. This is mainly done through the use of JavaScript features such as AJAX and direct modification of the document object model (DOM). The system's front-end is mostly developed using JavaScript (accounting for roughly 80% of the codebase), using the latest standard of ECMAScript (ECMAScript 2017). As such, the remainder of this section will be discussing the JavaScript applications architecture.

In recent years, modern websites have become more dynamic or "desktop-like", and are starting to replace many traditionally desktop-based applications, such as word processors or accounting systems. This phenomenon is commonly referred to as web 2.0 and can be seen across all fields of software. These websites, often referred to as web applications, are mainly dependent on JavaScript to make the user experience fluent and

without hiccups (e.g., removing the need for page refreshes), resulting in monolith sized applications written purely in JavaScript.

In Mimir, we use the web interface library ReactJs (React) [31] to achieve this "desktop-like" environment. React is a user interface (UI) library developed and maintained by Facebook, which has been growing in popularity ever since its release a little over five years ago (March of 2013). It uses a concept known as the virtual DOM to quickly update the contents of a web page. Choosing a JavaScript library/framework as a base of our application is an important decision as it will have profound effects on how we organize our project and may potentially limit the compatibility with other libraries. At the time of starting Mimir, we limited our choice of framework down to three potential candidates; React, Angular, and Vue [31, 38, 100]. Of which, React, and Angular have undoubtedly the largest market share, with Vue quickly gaining traction within the web development community. We decided to use React because of its JavaScript centric design, significant market share (compared to Vue), and relatively light size (file and feature wise) compared Angular. One could argue why we decided to use a framework at all, as our codebase is relatively small and could probably be developed without the need of any of these frameworks. This decision mainly came down to the ease prototyping and ingrained structure imposed by using such frameworks. At the start of Mimir, we did not know exactly how extensive the application would be, meaning we did not want to have to migrate our application mid-way through development, so we decided to play it safe an start using a framework from the beginning. Additionally, our sole developer was already familiar with the frameworks in use, alleviating us from the learning curve associated with using new technology.

From a front-end architectural point of view, Mimir is implemented using the Flux pattern [29], which in recent years has become popular among web applications. Unlike the more traditional model-view-controller (MVC) [67], flux uses a unidirectional data flow, making the application state and data flow less complicated and easier to reason about. In general, Flux consists of three major parts; a dispatcher, stores, and the views. As a user interacts with the web-interface (view), the interaction triggers an action, of which the dispatcher signals various stores to update the application state, which in turn, is reflected in the affected application's interface. A basic illustration of the general Flux process is shown in Figure 3.2, which shows the underlying agenda of how state moves in a single direction as denoted by the directed vertices. In Mimir, this pattern is implemented using a library called Redux [80], which simplifies the Flux pattern only to use a single store (meaning all application state is located in the same place). This library goes hand-in-hand with React, so it was a natural choice.

As a means to test the web application, Mimir includes an automatic test suite implemented in Jest and Enzyme [2, 30]. Jest is JavaScript testing framework developed and maintained by Facebook and works with React

out-of-the-box. Enzyme is a testing utility, specifically used to aid in testing React components. Automated tests are essential for any application, but as Mimir is marketed as opensource, tests are especially important as contributing members need an easy way to see if their contributions break anything in the main codebase.

3.1.2 Back-end Architecture, Tools, and Technologies

Mimir's back-end (server) is a python based representational state transfer (REST) [81] API, implemented using the microframework Flask [34]. Flask is a microframework based on the Werkzeug toolkit and Jinja2 templating engine. It is referred to as a microframework because it aims only to include the core features of a modern web framework, and instead of bloating it up with potentially useless features, make it easy to extend through the use of first and third-party plugins/extensions. In contrast, the typical "fully-fledged" framework used for python development is Django, which comes out-of-the-box loaded with multiple features such as an administration panel, database abstraction through a built-in object relational mapping (ORM), and web templating engine to name a few. Choosing Flask over Django comes from a minimalist point of view, as we would rather begin small and build towards something bigger. Additionally, Mimir was never intended to be a production-ready product at the end of this thesis, so using such a "heavy" framework would most likely get in the way rather than save us time. For the most part, Mimir's server is used to interact with the underlying neural network. Mimir uses a SQLite database to hold necessary information about uploaded images and CNN models.

As previously discussed, Mimir is structured as a REST API, meaning it exposes simple stateless endpoints for interaction with various parts of the system. The advantage of using a REST based architecture is the complete separability between server and client, meaning one could easily separate the two and only use the API exposed by the server. As of now, the endpoints exposed by Miimr fall into one of three categories:

- Endpoints related to uploading, modifying and deleting images and videos from the system.
- Endpoints related to uploading, modifying and deleting CNN classification models to be used for multimedia analysis.
- Endpoints related to the analysis of multimedia.

Images and videos uploaded to Mimir are stored in the SQLite database together with an identifier and basic meta-information. As videos are uploaded, they are automatically split into individual frames and stored together with the uploaded images. The identifiers are later used for retrieval when requesting an analysis, which takes the form of either classification or the generation of various visualizations. Classifications

and visualizations are stored together with their respective image as requested. The CNN models used for analysis follow a similar pattern to that of the multimedia endpoints. Using the API, one can upload, modify, remove and selected models for use in Mimir. Analysis endpoints allow for the classification and visualization of various layers of a selected neural network. Additionally, it exposes an endpoint for a full analysis, which bundles all forms of classification and visualization into a single request.

3.1.3 Deep learning Tools and Technologies

As the popularity of deep learning has increased over the last few years, so has the number of deep learning libraries and tools used to aid in the development and implementation of these algorithms. Some of these libraries are better suited for specific scenarios, such as targeting particular architectures or distinct problem sets. Some libraries even come with pre-trained networks, making it very easy to get started using various popular neural network architectures. In this section, we will give a brief overview of some of the most prominent deep learning libraries commonly used in real-world applications and research. Please note that there seems to be some confusion surrounding the terms library and framework when it comes to the description of these deep learning technologies, so for consistencies sake, we will be referring to all the described technologies as libraries unless explicitly stated otherwise.

As we have previously discussed, training a deep neural network is computationally expensive, and may take a long time depending on the machine used to train it. The resurgence of deep learning in computer vision tasks is primarily due to the introduction of powerful graphics processing units (GPUs), allowing for high parallelization among its many neurons used for computing. GPUs are exceptionally well suited for deep learning as a typical neural network consists of layers containing many identical neurons which may efficiently be computed in parallel. Additionally, there are other benefits such as a higher memory bandwidth when compared to a central processing unit (CPU). This is an important consideration when choosing a deep learning framework, as not supporting GPUs would result in significantly longer training time. Thankfully, most libraries now fully support GPUs, although they are typically aimed towards Nvidia produced GPUs, with few openly supporting AMD GPUs.

In an effort of making the deep learning accessible and easy to use, yet still, keep it computationally efficient, most deep learning libraries utilize an interface language, often called host language, different from the underlying implementation. This means that much of the mathematical complexity is abstracted away, and the researcher can entirely focus on the implementation of the network in question, without having to worry too much about the efficiency of the mathematical operations. The host language is generally implemented in a high-level language, most

Library	License	Interface	Opensource	CPU	GPU	Graph
Tensorflow	Apache 2.0	Python, C++	✓	✓	✓	Pre-built
Torch	BSD license	Lua, C	✓	✓	✓	Pre-built
PyTorch	BSD license	Python	✓	✓	✓	Dynamic
Caffe	BSD license	Python, C++, MATLAB	✓	✓	✓	Pre-built
Theano	BSD license	Python	✓	✓	✓	Pre-built
Deeplearning4j	Apache 2.0	Java, Scala, Clojure, Kotlin	✓	✓	✓	Pre-built
CNTK	MIT license	Python, C++	✓	✓	✓	Pre-built
Keras	MIT license	Python, R	✓	✓	✓	Pre-built

Table 3.1: A brief overview of some of the most prominent deep leanring libraries as of 2018.

commonly Python, making it accessible among scientists who may not be programmers by trade.

Most deep learning libraries utilize symbolic computing, meaning the host language describes an underlying computational graph, which is the compiled and executed. A comprehensive look at computational graphs is outside the scope of this thesis, but in simple terms, a computational graph is a directed graph where the edges correspond to either operations or variables. This allows for efficient use of memory and fast execution as all operations are known before runtime, in contrast to imperative programs where one must account for all possible outcomes. The downside of such an architecture is that it must be setup before execution, not allowing for the dynamic execution of code as the graph is being computed. This has partly been solved through the use of a dynamic computational graph [65], implemented in various libraries such as PyTorch [74].

Most deep-learning libraries come with pre-built and pre-trained models, which may be used straight out-of-the-box. This is useful for the most common computer vision tasks which relate to objects found in everyday life, e.g., cats, dogs, flowers, etc. But when it comes to medical imaging data, one can assume that these models won't work quite as well and need some form of tuning before being served. This may be in the way of training a network from scratch or using transfer learning (explained in Section 4.1).

As previously mentioned, there are plenty of deep learning libraries on the market today. Looking to choose a library which fit our needs, we decided to narrow the selection down to the most prominent libraries on the market today. The libraries we considered were Tensorflow, Torch, Pytorch, Caffe, Theano, Deeplearning4J, CNTK, and Keras [12, 15, 74, 90, 101, 103]. A brief over of each library can be seen in Table 3.1. In the end, we decided to use Keras as merely because of its simplicity. Keras comes with several pre-trained CNN architectures, making it quick and

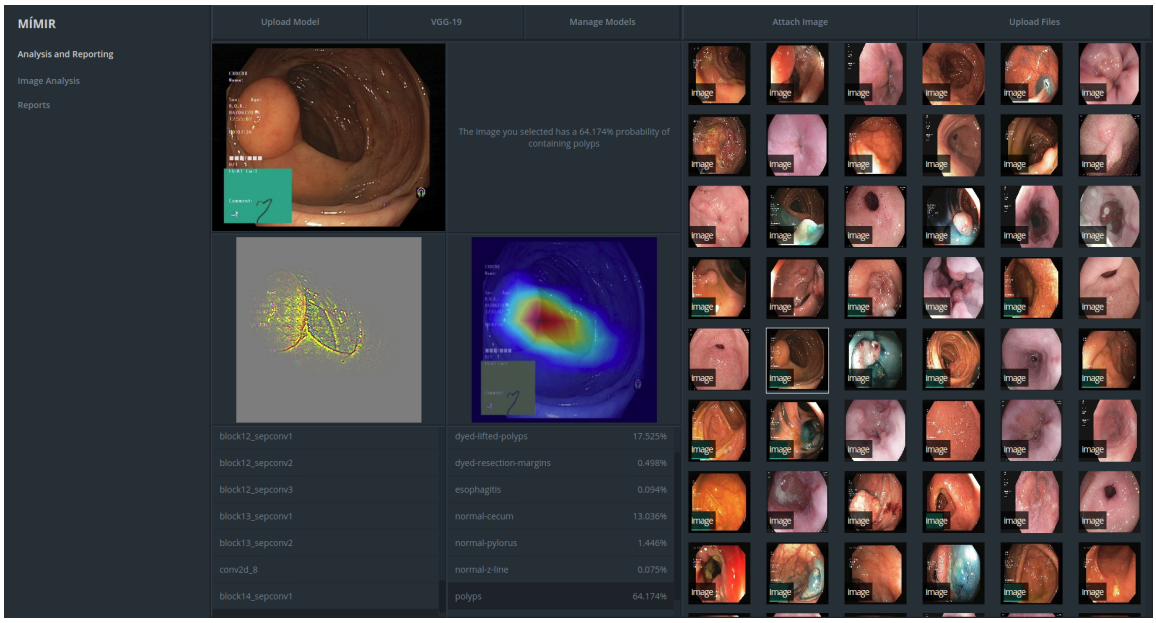
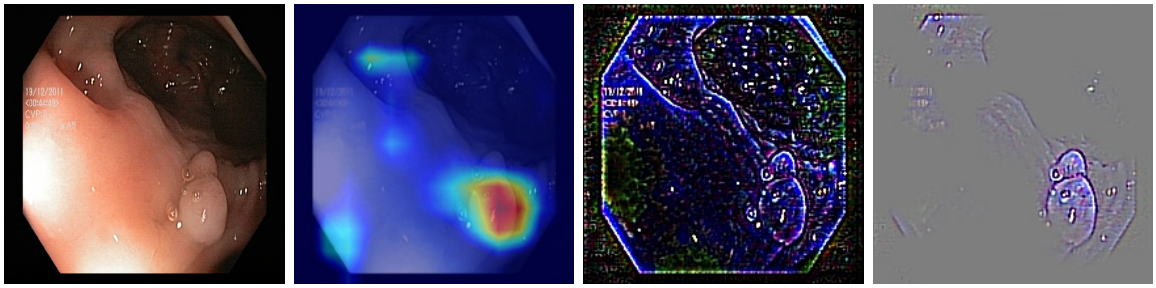


Figure 3.3: The web based user interface of the neural network dissection tool included in Mimir.



(a) Original image (b) Grad-CAM (c) Saliency Map (d) Grad-CAM

Figure 3.4: Image representations used by Mimir to explain the internals of a deep convolutional neural network.

easy to get started experimenting with various deep neural networks. In addition to this, Keras also includes various tools for data preparation and data generation, making training a neural network from scratch (or through transfer learning) relatively easy. Although the base API of Keras is quite simple, it still allows access to the complexity of the underlying back-end. This is done through Keras's *back-end*, which translates the called operations into the used back-end counterparts. We use the Keras back-end API to produce the various visualizations of Mimir, which we will go into more detail in Section 3.2.

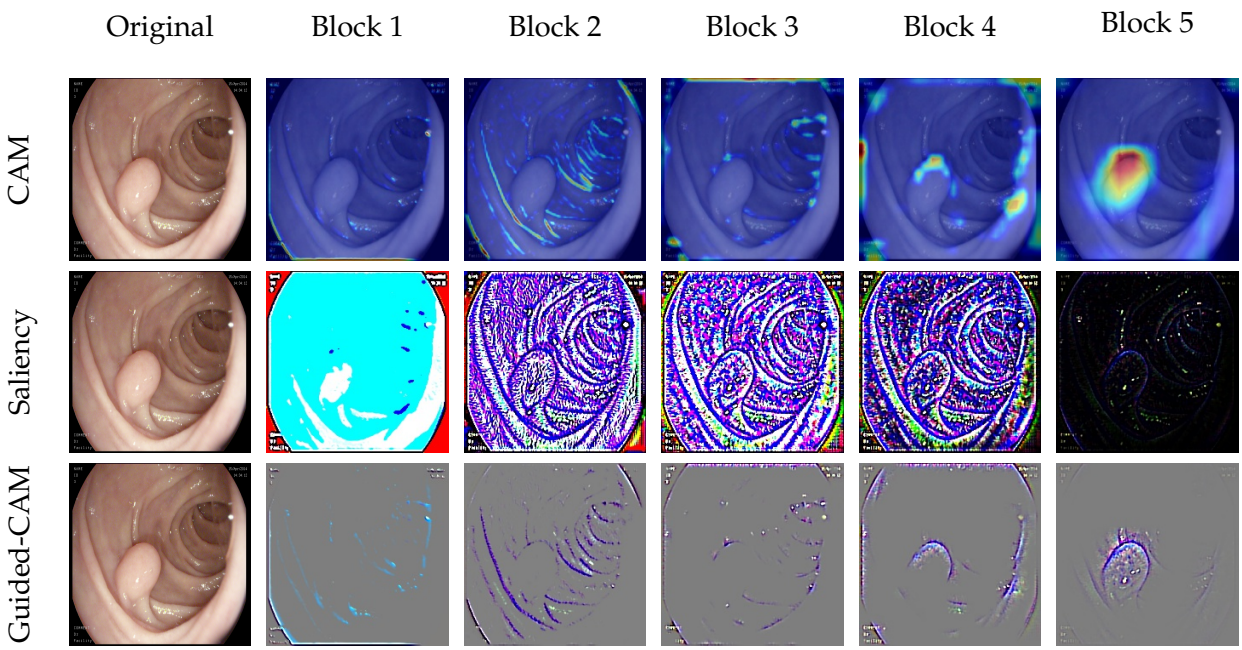


Figure 3.5: An image of the class "polyp" being visualized by a VGG-19 based model at the last layer of each convolutional block. Note that the saliency image is not presented in Mimir, but generated as an intermediate step to produce the guided-CAM.

3.2 Neural Network Dissection Tool

As we briefly discussed at the start of this chapter, part of the Mimir's objective is to make the analysis of deep CNNs transparent enough that a non-technical user may be able to understand what features of a given image lead to the model suggesting a diagnosis. This phase is supported by the neural network dissection tool, which attempts to demystify the inner workings of a deep CNN by presenting a variety of visual representations of the given image, seen as by the network, as the image moves through its various layers. Each of visualization gives a slightly different perspective of what the network is "seeing" at a given point the network (layer) for a target class. Through the use of this tool, the user may verify that the object itself signals an object detection, and not by noise or other artifacts. This is not only useful for medical doctors who need additional background to give a thorough diagnosis but may also be handy for researchers needing a way to diagnose issues within a trained CNN, of which the tool may give pointers to additional pre-processing steps that might lead to better results (classification performance and generalizability). An example of the three visualizations generated for a given image is shown in Figure 3.4, where we see the original image (Figure 3.4a) together with the grad-CAM (Figure 3.4b), saliency map (Figure 3.4c), and guided grad-CAM (Figure 3.4d) visualizations.

The tool works for both images and videos, with videos being split into individual frames and processed one by one, in the same way as for single images. In its current state, the only way to get data (images or video) into the system is by manually uploading content through the upload button located in the upper right corner of the UI (as seen in Figure 3.3). In a deployed system, it would be natural that frames would be captured directly from an ongoing colonoscopy through a live video stream in addition to this manual option.

As an image or video is uploaded to the system, it is automatically scanned and classified into the categories of the underlying CNN. The user is then presented with the uploaded material as seen in the right pane of the UI (Figure 3.3), and can from here select an image/frame for further analysis. Note that each image in this panel has a label, this label corresponds to the classification it got under analysis. Upon selection, visualizations are generated on the fly based on the selected image, a target layer, and a target class (target layer and target class have a default value if not selected). Once the visualizations are complete, the user is presented with the predicted image class, a grad-CAM visualization of the image, a guided grad-CAM visualization of the image, a list of the convolutional layers corresponding to the selected CNN, and a list of possible categories accompanied with each respective probability. From here, the user can select different target classes and target layers to generate further visualizations. Selecting different target classes may be useful when there are traces of multiple classes within the same image. By targeting different classes, the user can view what regions of an image directly correlate to the predicted output of the system. For example, if a polyp is discovered near the cecum, the system is likely to give a relatively high probability for both the polyp and cecum class. By targeting each of these classes, one will be able to see which regions of an image directly correlate to the given probability (hopefully the area surrounding the polyp), and which areas correlate to the cecum probability. For layer selection, the system defaults to the last convolutional layer of the CNN, showing what the network recognizes right before it makes its prediction. For the most part, this is what we want. But it may also be useful to look further back in the network to see what less abstract features are detected in previous parts of the network. An example of this is shown in Figure 3.5, where we visualize an image containing a polyp at the last convolutional layer of each convolutional block of a VGG-19 CNN. Looking at the last convolutional block in the figure, we see that the network correctly detects the polyp located in the central region of the image, verifying that the network at the very least has some notion of what features relate to polyps. Please note that the saliency map is not viewable from the Mimir's UI. We decided to exclude this representation as it did not add any meaningful information to the classification process not already present within the grad-CAM and guided grad-CAM representations. It is however needed to produce the guided grad-CAM visualization, so it still plays a pivotal role in the generation of these visualizations.

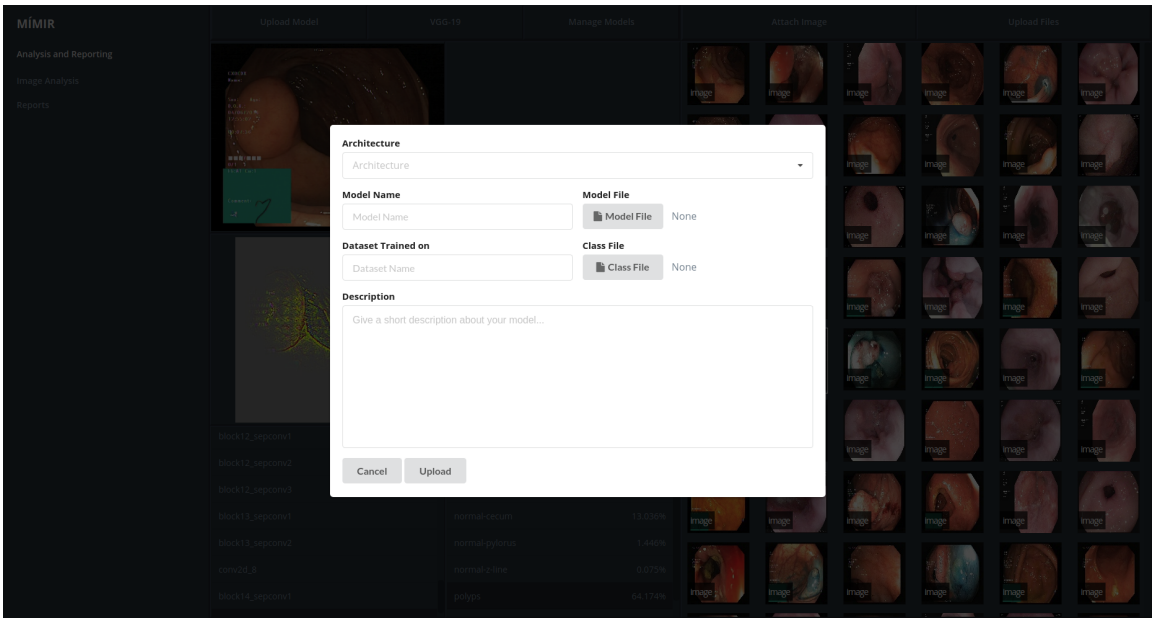


Figure 3.6: The upload dialog presented to the user when uploading new models to Mimir. Please note that it requires both a model file and class file, with the model file being a Keras HDF5 model file, and the class file being a JSON file foramtt as seen in Figure 3.7.

```
{
    "0": "dyed-lifted-polyps",
    "1": "dyed-resection-margins",
    "2": "esophagitis",
    "3": "normal-cecum",
    "4": "normal-pylorus",
    "5": "normal-z-line",
    "6": "polyps",
    "7": "ulcerative-colitis"
}
```

Figure 3.7: An example of the contents of a class file, formatted as required by Mimir for uploading new models to the system. Please note that the file is in complience with the JSON standard, using a key-value store to tie relations between class id and label.

In its current state, Mimir has official support for a variety of standard CNN architectures, including VGG-16, VGG-19, Inception (v3), ResNet-50

and Xception. For this work, we focused on this small list of architectures as we needed a starting point for the system. Although the official support is limited to the models described, it should work on any CNN architecture as long as it contains convolutional layers. To add additional models, one can manually upload a Keras HDF5 model file, which will automatically be added to the system's database and ready for immediate use. This feature is currently limited to Keras HDF5 files, as our implementation is based on this assumption. To upload new models to Mimir, one starts by clicking the "Upload Model" button located in the upper right part of the UI. This brings up a dialog (shown in Figure 3.6) where the user can set some meta-information about the model, such as its name, its base architecture, the dataset used to train it, and a brief description noting other features one may wish to include. In addition to a model file, a class file in the form of a simple key-value JSON file is needed in order to match labels to the given prediction of the model correctly. This JSON file must list the index and label in key-value pairs as shown in Figure 3.7. After a model has been successfully uploaded to the system, one can manage all uploaded models through the model manager (shown in Figure 3.8), which is opened through the button labeled "Manage Models" located to the right of the "Upload Model" button. Here the user sees a list of all uploaded models and has the option to activate, modify or delete models from the system. Upon activating a model, it is automatically loaded by the server and ready to use once the UI has finished loading. One can verify what model is selected by looking at the title located between the two before mentioned buttons, which may also be clicked to get a brief overview of the currently active model. As one may think, this feature is not intended for medical doctors or non-technical end users. It is mostly targeted towards researchers or other professionals testing various models without having to reload the system every time one wants to switch model. Additionally, it played an essential role in our experiments when diagnosing issues with the Kvasir (v2) dataset, which will be covered in Chapter 4.

Now that we have a good understanding of how the neural network dissection tool is used, we will in this section take a more detailed look at the generated visualizations used to give context to the prediction of a deep CNN. As we previously mentioned, the three visualizations generated by Mimir is a grad-CAM, saliency map (made through guided backpropagation), and guided grad-CAM. Of which, the grad-CAM and saliency map is created independently from each other, and the guided grad-CAM being a combination of the two. We use the guided grad-cam representation together with a grad-cam to give two perspectives on what the CNN is "seeing" when making its prediction, which in turn will hopefully distill a higher amount of confidence in the correctness of the network in use. Principally, the grad-CAM and guided grad-CAM show the same information, albeit the guided grad-CAM includes a bit more detail, we decided to include both as the grad-CAM may be more evident in its explanation. The overall visualization process can be seen in Figure 3.9, and the following explains it in a bit more detail.

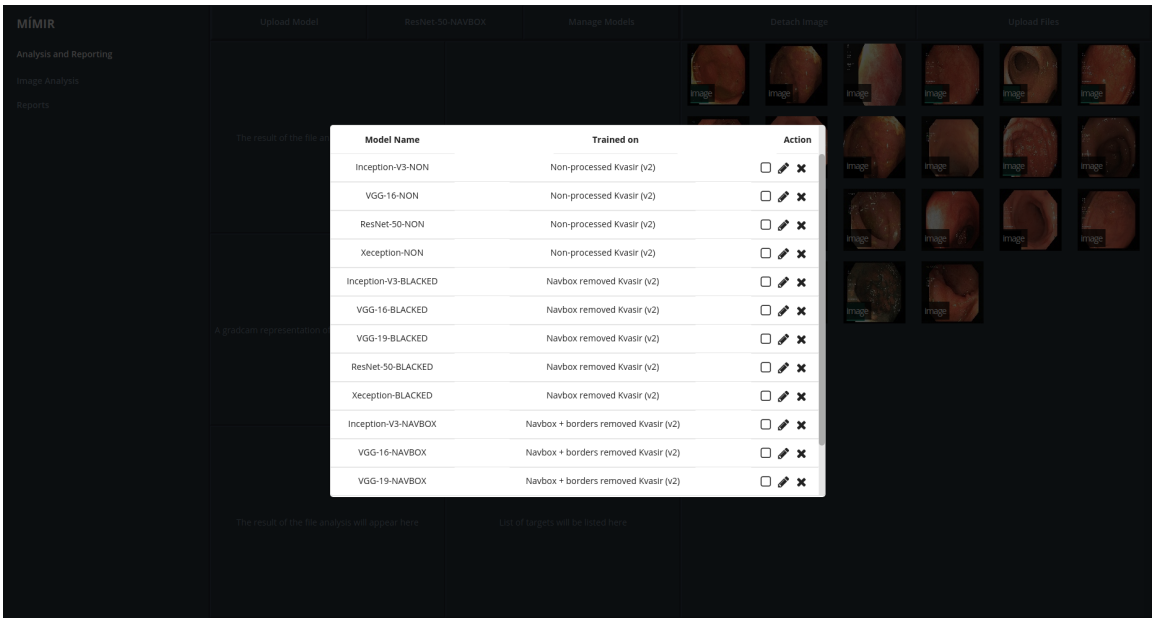


Figure 3.8: Mimir includes a model manager for managing all uploaded models. From here the user may activate, modify or delete any previously uploaded model.

The visualization process starts once the user has selected an image, layer, and class for further inspection. With an image, target layer and target class chosen, we calculate the gradient of the target layer using the loss of the target class in regards to the image. These gradients are globally average pooled to get the weights, which is multiplied by the output of the target layer and passed through a ReLU function to produce the grad-CAM. The grad-CAM is then re-sized back to the original dimensions of the image and has its values squashed between 0 and 1 before applying a blue-red heat map filter.

To generate the saliency map (done through guided backpropagation), we start by replacing the activations of our original network with a modified ReLU function. During backpropagation, a traditional ReLU would let all gradients whose inputs were larger than 0 pass. We change the ReLU by adding the additional rule of discarding all gradients that are below 0, thereby only back-propagating the positive influence on the activations. With this modified network, we calculate the gradients of the target layer with respect to the input image, i.e., these gradients represent our saliency map.

Once the grad-CAM and saliency map has been computed, we multiply them together to produce the guided grad-CAM representation. As one can deduce from this process, the grad-CAM representation is class specific visualization, which looks at what areas of an image activates the most based on the loss of a given class. The saliency map is not class specific, showing only what areas of the image have a positive influence on activation. Although not class specific, it gives a more detailed look at

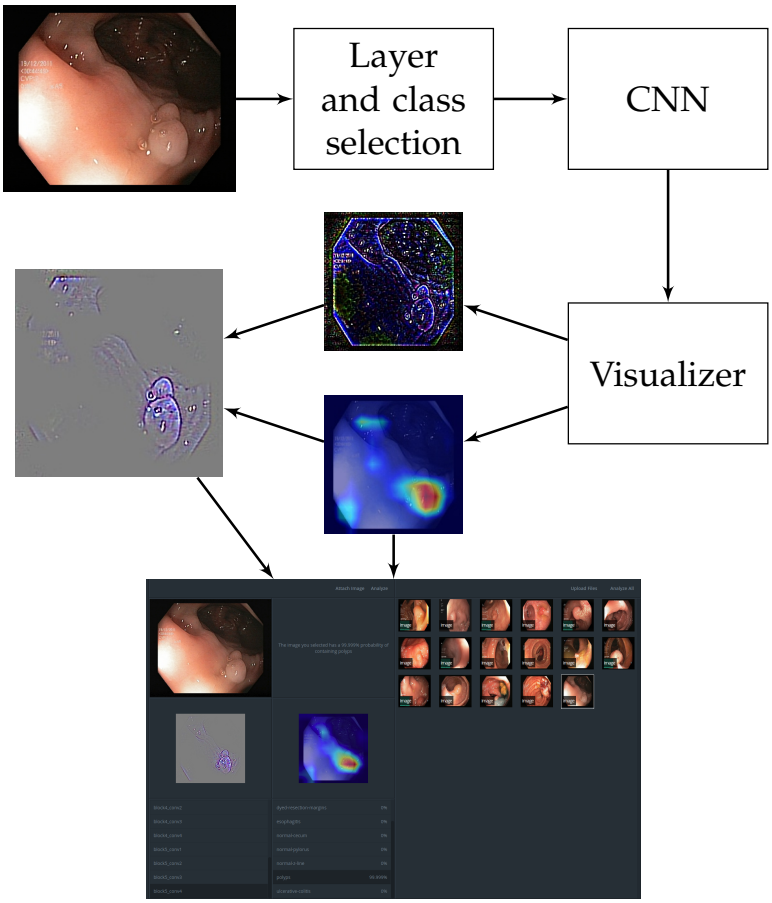


Figure 3.9: An overview of how we produce the two visualizations included in the image analysis, and how it is presented in the user interface where a visualization of the different convolutional blocks can be selected.

what features are being detected at any given layer. By combining these two visualizations, we can get the best of both representations, taking the class-specific properties of the grad-CAM and applying them to the pixel-detail features of the saliency map.

3.3 Report Generation Tool

To support documentation phase of a completed endoscopy, Mimir provides a basic tool for generating endoscopy reports. A screenshot of the tool can be seen in Figure 3.10, where we see the preview of a sample colonoscopy report produced by the system. As we discussed in Section 2.1.7, the quality of endoscopy reports is undoubtedly lacking, with many reports being submitted incomplete and with a lack of standardization [51, 61]. This tool aims to aid in this issue through the use of a WYSIWYG interface, where doctors can make direct modifications to the suggested report. At its current state, the report generation tool

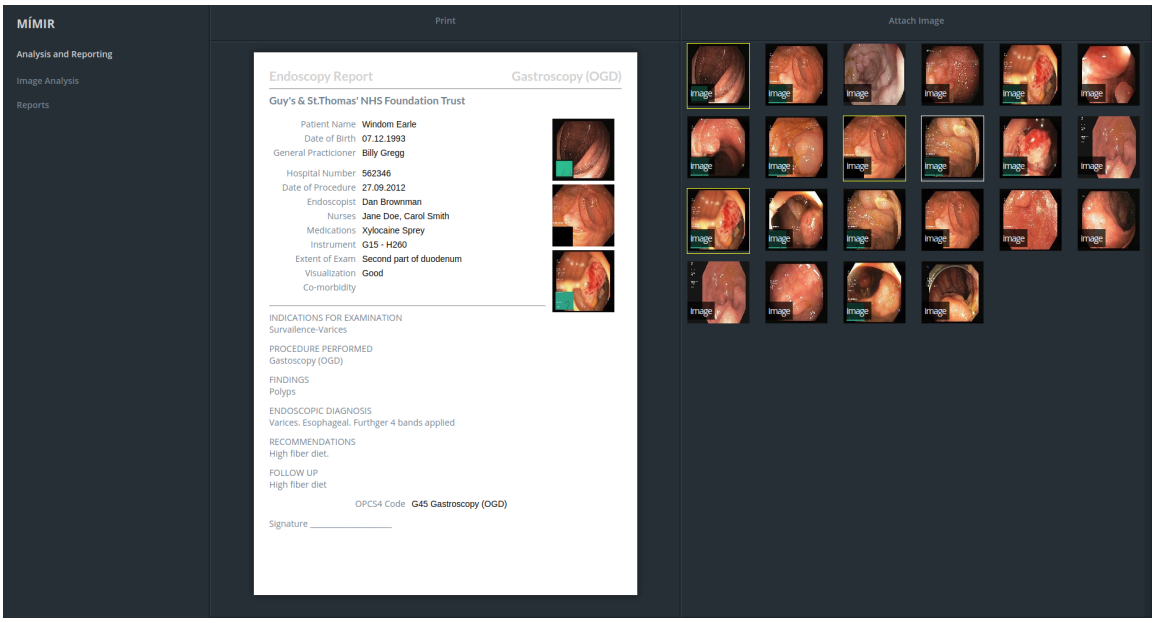


Figure 3.10: The web based interface of the report generation tool.

provides basic functionalities such as changing text and adding images through the image selector located to the right of the report in Figure 3.10. The images in this pane are sorted after highest probability of a given class, thereby having the system suggest which images are most relevant for a given diagnosis. The automatic suggestion of images is favorable as doctors mentioned that manually adding findings to a standard report generally takes about two minutes [40, 57]. But for complex reports, where many findings were found of which would have to be documented, reporting these findings could take 15 minutes or more to produce.

The need for automatic reporting systems based on automated multimedia analysis is essential in the improvement of documentation precision, standardization of report through the suggestion of terminology supported by the MST and world endoscopy organization (WEO) [1], and decrease the amount of time needed to produce complex reports. As the first iteration of Mimir focused on the understanding and interpretation of deep neural network analysis, this tool in its current form is a proof-of-concept. Plans for this tool include the automatic suggestion of text, based on the concluding analysis of a medical examination, support for multiple templates, and a variety of quality of life (QOL) improvements such as support for drop-down menus and drag and drop interfaces.

3.4 Use Case Scenarios

Below we have proposed a series of use case scenarios which we imagine Mimir could be a good fit. Scenario A, B, and C will focus on medical use cases, i.e., how a doctor may use this system. Scenario D and E will focus

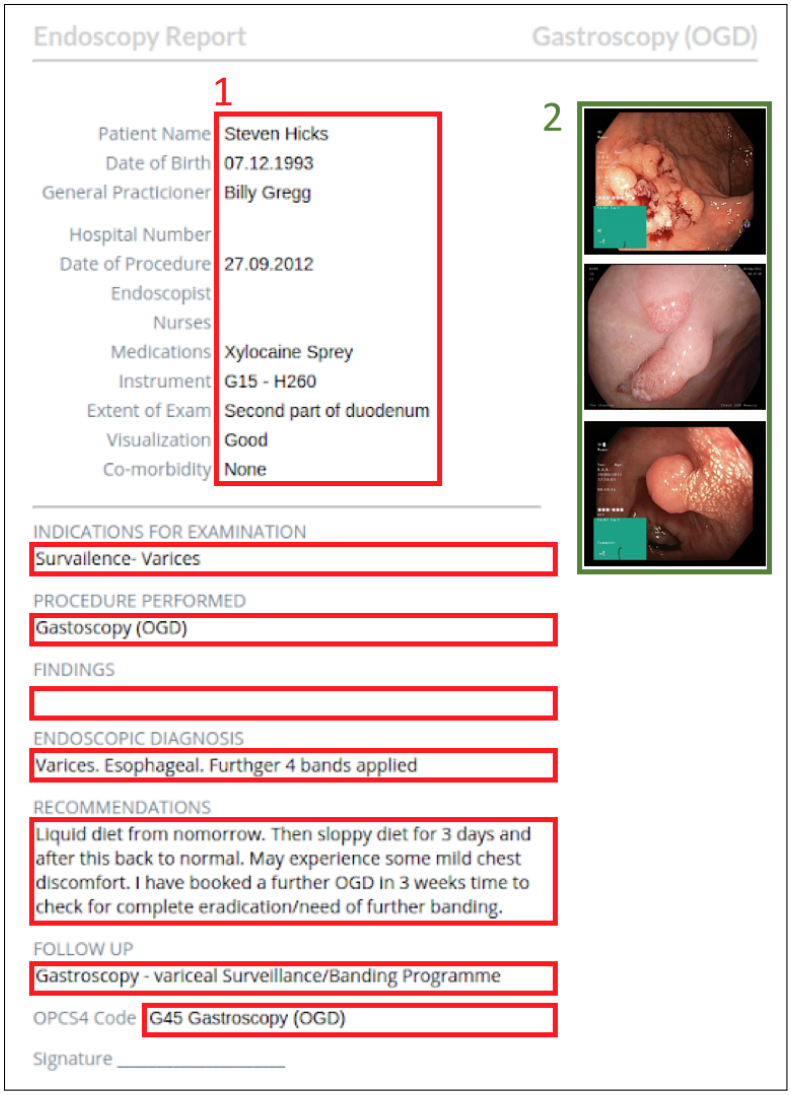


Figure 3.11: An example of an automatic generated report. The red area marked (1) shows the editable text fields. The green area (2) shows the images chosen for the report. Image taken from [46] and report based on sample taken from Wrestling the Octopus [41].

on a more neural network optimization point of few, i.e., how a researcher or scientists may use the system to improve trained CNNs.

Example Scenario A — Verify the Prediction of a Diagnosis After getting the diagnosis based on the analysis of the colonoscopy examination video, we would like to verify that the network does, in fact, detect the diagnosed abnormality presented. After the examination, the frames where abnormalities are detected are automatically presented to the user on the image analysis web-page. For a given frame, the user can look through the network and verify that the network does, in fact, detect the abnormality related to the diagnosis.

Example Scenario B — Getting Relevant Images for Documentation After a successful colonoscopy, a doctor is presented with the produced image data, already analyzed and suggests a diagnosis of Polyp. As a doctor, one would like to quickly find the images that correlate the most to the diagnosis and add them as attachments to the endoscopy documentation. Through the use of Mimir, one can quickly sort images by order of correlation (highest probability) and pick among a selection of suggested images. The selected images will automatically be added to the report in the form of image attachments. In its current form, Mimir allows up to five images attachments for any given report.

Example Scenario C — Generating a Colonoscopy Report After a successful colonoscopy, the video produced is automatically passed through the system and analyzed for abnormalities. Based on the diagnosis, the system would present images that support the diagnoses (which can be further examined as described above in Section 3.4). This would save the user time by not having to screen the frames of the video for the diagnosed abnormality and manually select image candidates.

Exampled Scenario D — Comparing Different Models After having trained a variety of models on the same dataset, a scientist wants to see which of the trained models has the highest chance of generalizing to unseen datasets apart from the evaluation set used under training. This could be useful as it is not always easy to gather large datasets for any given problem, meaning that the evaluation set may not be a good enough indication of whether or not the model will generalize well to data outside the current dataset. The scientists upload the models to Mimir and analysis which models seem to have correctly learned the features of the categories in question. Through this analysis, the scientist can identify which model learned the features of each class, thus has a higher probability of classifying unseen images of such classes in new, unseen images.

Exampled Scenario E — Determine which Pre-processing Steps to Apply Before Training After having trained a variety of models on a given dataset, a scientist is not happy with the general performance of each model. By uploading the models to Mimir, he/she can analyze the layers of the neural network to potentially find incorrect activations for a given task, such as activations related to noise or other artifacts. After discovering that a given class activates on a given artifact, e.g., reflections, the scientist can apply additional pre-processing steps the dataset before training, thus hopefully making the network learn the intended features, instead of the artifacts.

3.5 Summary

In this chapter, we presented the work of developing Mimir, an automatic reporting system with a focus on the trust and understanding of the applied deep learning algorithms. The purpose of this system is to aid medical doctors in producing multimedia-enriched reports through the aid of deep neural networks. An essential piece of this process is making the analysis done by the CNN understandable and interpretable among non-technical users, e.g., medical doctors. This is motivated by the need for transparency and understanding when it comes to diagnosing a patient with any serious disease. We achieve this through a variety of visualizations generated by the system, which allows the doctor to verify a diagnosis set by the Mimir. The development of this system relates back to the first two research objectives. Where we stated the need for a automatic disease detection system, whick gave background into why it produces a given result. Each of the first two objectives is supported by their own respective tool as part of the system.

With all system requirements met with the development of Mimir, we use this system to conduct a series of experiments to verify how it could be used to improve the performance of various models based on different architectures. This relates to our third research objective, where we wanted to test how this new found understanding of how a CNN classifies an image could be used to increase the performance of existing models. The method behind each experiment, together with their results, will be explained in the next chapter.

Chapter 4

Case Study on Mimir for use in Classification Understanding

As algorithms based on deep learning techniques are setting record high-performance levels across a vast number of fields, it is only natural that these algorithms would be applied to mission-critical areas as well. This, however, has brought up an important issue. Even though these algorithms show an improvement over the state-of-the-art, they are typically not trusted among the practitioners of these high-risk areas, such as doctors, as there is no tangible way of understanding why these algorithms work, i.e., what ground their output is based on. As we discussed in Section 2.2.4.1, the CEHC board decided against using neural network based algorithms for treatment against pneumonia as they were simply too hard to understand, even though they showed better performance than their counterparts. These methods have only become more complex with time, going from simple three-layered neural networks containing a countable number of parameters, to networks consisting of hundreds of layers.

In this chapter, we look at using Mimir for the analysis of deep CNNs with the purpose of discovering potential methods of improving the quality of models trained on the Kvasir (v2) dataset. Although initially meant for a medical audience, we found that Mimir could be a useful debugging tool for researchers and scientists developing deep CNNs, or determining what pre-processing steps could potentially lead to higher quality training data for a given dataset. To put our system to the test, we trained a variety of CNNs and analyzed their layers using Mimir's neural network dissection tool to find out why the trained models confuse certain classes, and what we could do to rectify this. Before presenting our results, we first look at the various networks and datasets used to train and evaluate our system. We then discuss the training and evaluation procedure for each model, showing what hyperparameters were utilized under training and how each model was assessed. With the initial results in place, we describe the process of using Mimir to analyze each model

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
VGG16	528 MB	0.715	0.901	138 357 544	23
VGG19	549 MB	0.727	0.910	143 667 240	26
ResNet50	99 MB	0.759	0.929	25 636 712	168
InceptionV3	92 MB	0.788	0.944	23 851 784	159
Xception	88 MB	0.790	0.945	22 910 480	126

Table 4.1: The various pre-trained models supplied by Keras [12] [1], evaluated on the ImageNet ILSVRC validation dataset.

and show the pre-processing steps implemented before re-training the models using the modified datasets. Finally, we will present our results and confirm whether or not the analysis gathered through Mimir helped us build higher models of higher quality.

4.1 Training, Datasets and Architectures

As we discussed in Chapter 3, Mimir supports a variety of CNN architectures, with official support for VGG-16, VGG-19, Inception (v3), ResNet-50 and Xception. To gain a good understanding of how a CNN trained on the Kvasir (v2) dataset "sees" a class, we decided to test each of the supported networks through the use of Mimir's neural network dissection tool. This, however, requires a training and evaluation strategy of how we may compare networks and dataset pre-processing steps. In this section, we give a brief description of each supported architecture, a look at the datasets which played some part in the training and evaluation process, and finally discuss how each network was trained and evaluated.

4.1.1 Architectures

Since the introduction of convolution based neural networks in 1998s [58], many CNN architectures have emerged, with most using different techniques to create "deep" models without being too computationally expensive. Below we present some of the most common architectures within the field of image classification, describing their structure and what makes them unique. Each of the described networks is implemented in Keras, of which we used the standard configuration. Table 4.1 shows some necessary information about each of the trained models, giving some insight into how they differ on a high-level.

[1] https://keras.io/applications/

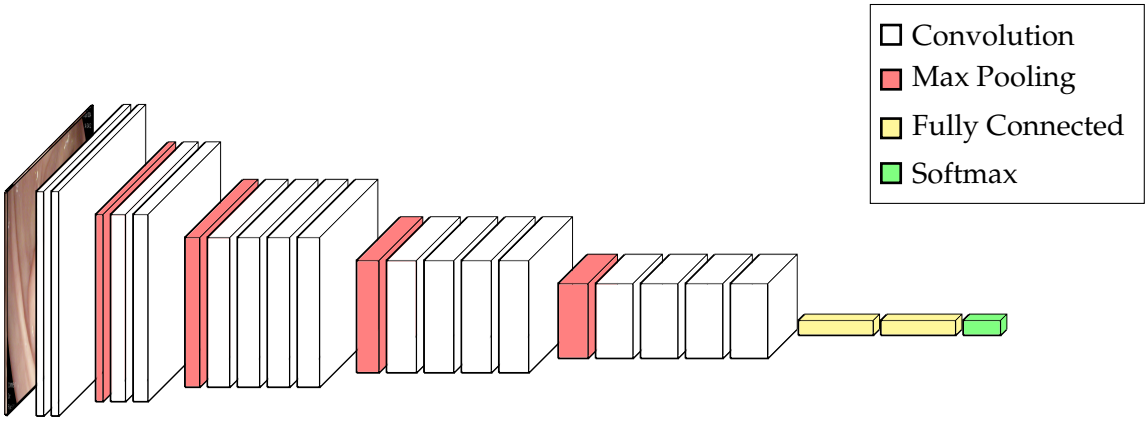


Figure 4.1: A visual representation of the VGG-19 architecture, with the different layer types color coded. White layers are convolutional layers, red layers are polling layers, yellow layers are fully connected layers and the green layer is the softmax classification layer. Note that the width and depth of a layer is reflected in the spatiality (width and height) and depth of each layer.

4.1.1.1 VGG Architectures

The VGG neural network[2] architecture describes a set of straightforward models which were introduced by Simonyan *et al.* in their 2014 paper "Very Deep Convolutional Networks for Large Scale Image Recognition" [97]. The network is built up by a series of five convolutional blocks with increasing depth, with each block being followed by a max pooling layer to reduce dimensionality. Each convolutional block contains some 3×3 convolutional layers, with the number of layers being dependent on the implemented configuration. The network ends with two fully connected layers with each 4096 nodes before softmax is applied for classification.

In its standard configuration, the network expects an RGB image with the size 224×224, but this may change depending on how the network is trained. VGG architecture comes in a variety of configurations, each with a different amount of weighted layers. The most prevalent of these is the VGG-16 and VGG-19 architecture containing 16 and 19 weighted layers respectively. Figure 4.1 shows a visual depiction of a VGG-19 network, displaying how the network becomes smaller in width and height, yet more in-depth, as the blocks progress. The VGG architecture has become popular for its simplicity and its high performance. In 2014, it took first and second place in the ImageNet Challenge for the localization and classification tracks [97].

[2]VGG is an acronym for "Visual Geometry Group", which is the group that submitted the network to the ILSVRC in 2014.

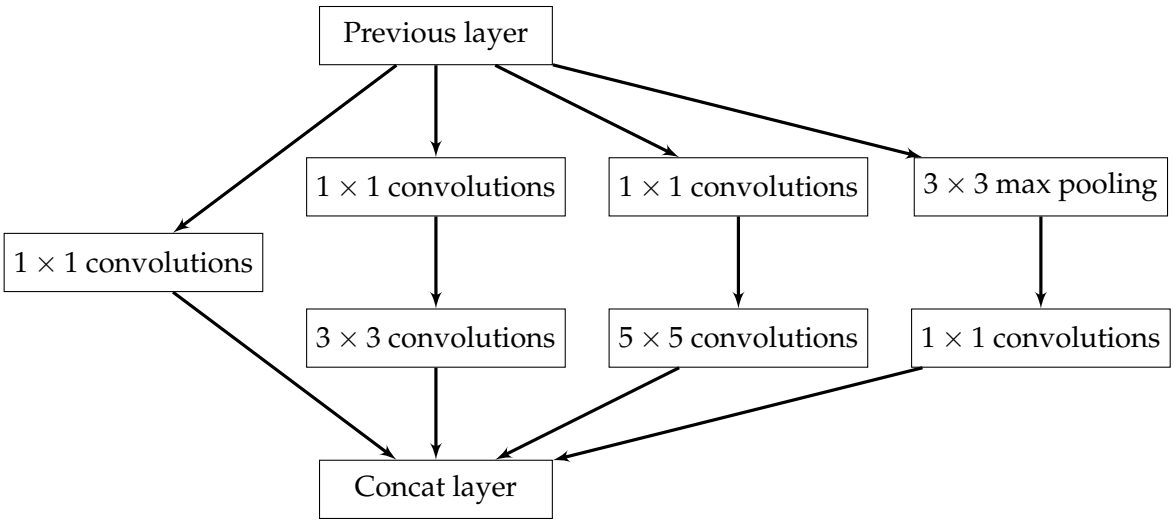


Figure 4.2: A visual example of the Inception module, used extensively in Inception based models.

4.1.1.2 Inception Architectures

The simplest way to improve the performance of a deep neural network is by increasing its size. This can be done through either (or both) increasing its depth (number of layers), or increasing its width (number of units per layer). Although this might be the simple and straightforward way of improving the quality of a model, it comes with two significant drawbacks. Firstly, increasing the size of a network means increasing the number of parameters. This makes the network more prone to overfitting, especially if the dataset used for training is limited (which in the medical field, is common). The second drawback of increased network size is the increase in computational consumption. For example, if two convolutional layers are chained in a convolutional neural network, any uniform rise in the number of their filters results in a quadratic increase of computation. Since the limit of the computational resource will always be finite, it is preferred to keep the computational cost of a network relatively low while also improving its performance.

With the presence of these drawbacks, Szegedy *et al.* presented an architecture with the goal of reducing the computational cost of training large neural networks by decreasing its number of parameters [99]. The produced architecture was called Inception (based on the movie of the same name), because of its core building block, the inception module, is a mini neural network in its own right. The inception module stems from the question of which convolution should we use for a given layer. This is not always obvious, so the inception network lends this decision to the network itself. For each inception module, a series of convolutions are performed, e.g., 1×1, 3×3, and 5×5 convolution, in addition to a pooling operation. Then the output of all operations are concatenated, and we trust that the network figures out what information is useful. The

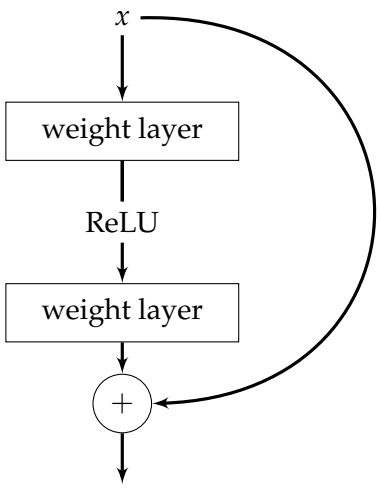


Figure 4.3: A visual example of the Residule block, used extensively in ResNet based architectures.

reason we perform a polling operation in addition to the convolutions is simply that high performing networks typically use polling operations, this is at least the purpose stated in the paper. Figure 4.2 shows such a module, where we see the three beforementioned convolutions and the polling layer. Additionally, we see that each of the convolutions is preceded by a 1×1 convolution (except for the 1×1). As we previously mentioned, part of the goal of the inception architecture was reducing the number of parameters in the network, adding multiple convolutional operations to each layer inevitably increased it. This is where the 1×1 convolution is handy. The 1×1 convolution is used to reduce dimensionality, thus reducing the number of computations performed in the larger convolution.

4.1.1.3 Residual Neural Network Architectures

Residual neural networks (ResNet) is a neural network architecture proposed by Microsoft as a solution to the issue of training very deep neural networks [42]. Before we explain how the ResNet architecture improved the training process of very deep neural networks, we will first present some background to why training very deep neural network was difficult. In theory, a deep neural network should always be able to perform as good, if not better, than a network containing fewer layers during training (when overfitting is not an issue). For example, a network containing $n + 1$ layers should be able to perform as well as a network containing n layers, even if only by copying over the same first n layers and performing an identity mapping[3] for the last layer. This, however, does not work in the real world. The first issue with training very deep neural networks is the problem of *vanishing gradients*. As explained in

[3] An identity mapping ensures that a node's input is equal to its output, in the context of a layer in a neural network there should be no change in the layers input as its output.

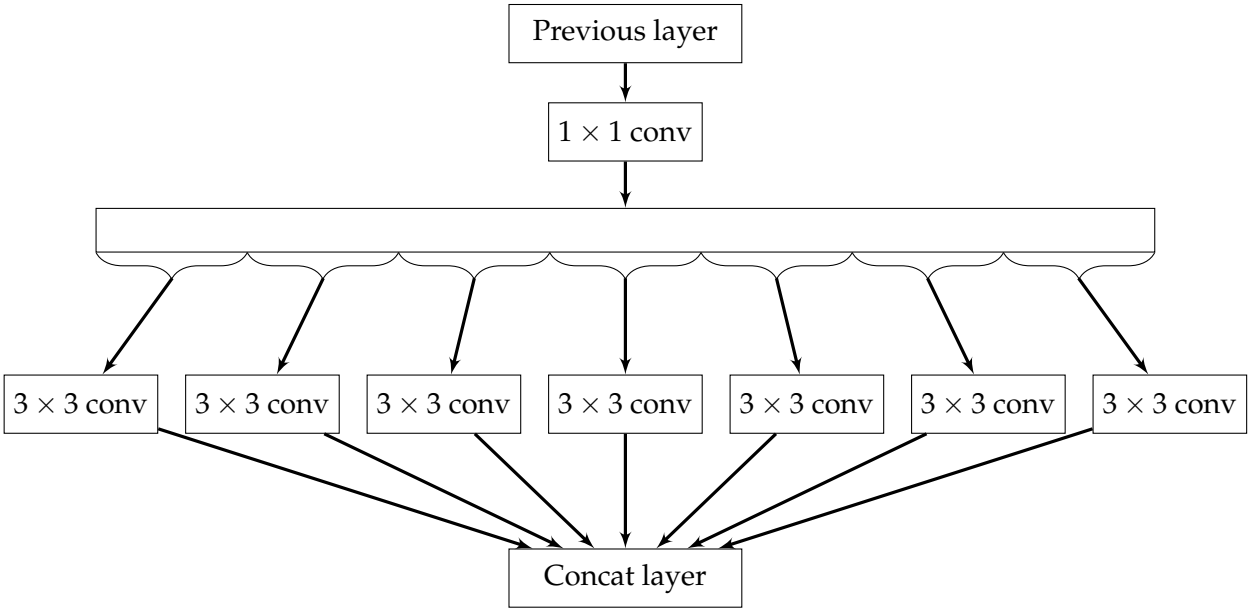


Figure 4.4: A visual example of the Xception module, used extensively in Xception based architectures. The illustration is based on the figure present in the original paper **??**

Section 2.2.2.3, neural networks learn by multiplying the gradient of its output with a learning rate. If the gradient becomes too small, then the network will stop learning. This becomes more of an issue the deeper the network gets, resulting in a hard time in making them converge. This issue was primarily solved using normalization layers, spread across the network, which made training deep networks to convergence a much easier task. However, this sheds light on another problem with training these deep networks. Network accuracy would degrade as network depth increased (commonly known as the *degradation* problem). This degradation problem is what He *et ai.* set out to solve when developing the ResNet architecture [42].

ResNet solves the accuracy degradation problem by instead of learning the underlying mapping from $x \rightarrow f(x)$, we learn the difference between x and $f(x)$. Then, to calculate $f(x)$, we can add the difference to the input. This process has implemented the use of so-called residual blocks, which looking at Figure 4.3 shows a 'shortcut' connection of adding the input to the output of the block. This technique led to the possibility of creating very deep neural networks that still performed well. The paper proposes networks of varying depths including 50, 101 and 152, with performance increasing with the depth. ResNet was used in the 2015 ILSVRC and comon objects in context (COCO), and secured 1st-place in all five of the main tracks.

4.1.1.4 Xception Architecture

Of the architecture discussed, the Xception architecture is the newest (although not by much). Xception stands for "extreme inception", which as the name suggests, takes the concept of the inception architecture to the extreme. The architecture was proposed by François Chollet, which is also the author of Keras, in his paper "Xception: Deep Learning with Depthwise Separable Convolutions" [13]. The Xception network is based on the hypothesis that one can assume that cross-channel correlations and spatial correlations can be mapped separately. What this means in practice is that unlike a standard convolution, where we simultaneously look at the spatial dimension and depth dimension, Xception separates the two operations by first performing a depthwise convolution and following this up with a pointwise convolution (this is further explained in Section 2.2.3.2). Intuitively, we can think of this as first looking for correlations in two-dimension space, followed by searching for correlations in the one-dimensional space. Performing a linear combination of these two operations, we can learn the mapping of all three dimensions. This type of convolution is referred to as a pointwise-separable convolution and is also found in the Inception architecture, albeit to a lesser degree. Figure 4.4 shows a visual example of the Xception modules, where we see how the input is first passed through a 1×1 convolution to map cross-channel (depth) correlations, then separately map the spatial correlations of every output channel. Comparing Figure 4.2 and Figure 4.4 we see that the Xception module is based on the dimensionality reduction ideas of the Inception module but to a more extreme degree.

4.1.2 Datasets

Training and evaluation of selected models involve various datasets. As previously mentioned, the networks are primarily trained on the Kvasir (v2). But as we are not training these networks from scratch, we are essentially re-training a network that was previously trained on the ImageNet dataset. Repurposing a model for a new task is called transfer learning, and has become a popular approach to training deep neural networks. Transfer learning can typically be split into three distinct phases;

1. Select a set of pre-trained weights corresponding to the selected model (or train one from scratch).

2. Remove the top layer of the network responsible for classification (typically softmax) and add a new one corresponding to the new problem set.

3. Optionally fine-tune the model by re-training parts of it (this is generally done if the re-purposed problem set is very different from the original).

Figure 4.5: Eight example images taken from the ImageNet image database. Notice that ImageNet contains a wide veriery of objects, ranging from different types of animals to everyday household objects [88].

The consensus behind transfer learning is that we can re-use the learned features from a pre-trained network, such as shapes and curves, and apply them to a new domain. If the problem area is very similar to the original, one may be able to get away with only re-training the layer responsible for classification (typically softmax), but this is usually not the case. This is where fine-tuning comes into play. Under fine-tuning, we re-train (on top of the original weights) a set number of layers going back from the output layer, e.g., the five layers before classification. The reason behind no re-training the initial layers of the network, is because as we move closer to the output, the network becomes increasingly abstract. This means that the initial layers typically learn fundamental features such as edges or lines, while latter layers frequently learn features related to the target set (such as the shape of a dog). This is why we generally have to re-train more substantial parts of the network when dealing with vastly different problem sets. In the following sections, we take a brief look at the datasets which served some purpose for this work.

4.1.2.1 ImageNet

ImageNet [88] is a large database of images designed for object recognition software. It currently consists of over 14 million hand-annotated images, categorized to over 20 thousand categories. Since 2010, ImageNet hosts a competition, called the ImageNet ILSVRC, where research teams can compete against each other to achieve the highest accuracy on several visual recognition tasks. For each competition, the ILSVRC provides a slimmed down list of categories of which the submitted algorithms will be evaluated. The validation dataset provided in the ILSVRC hosted in

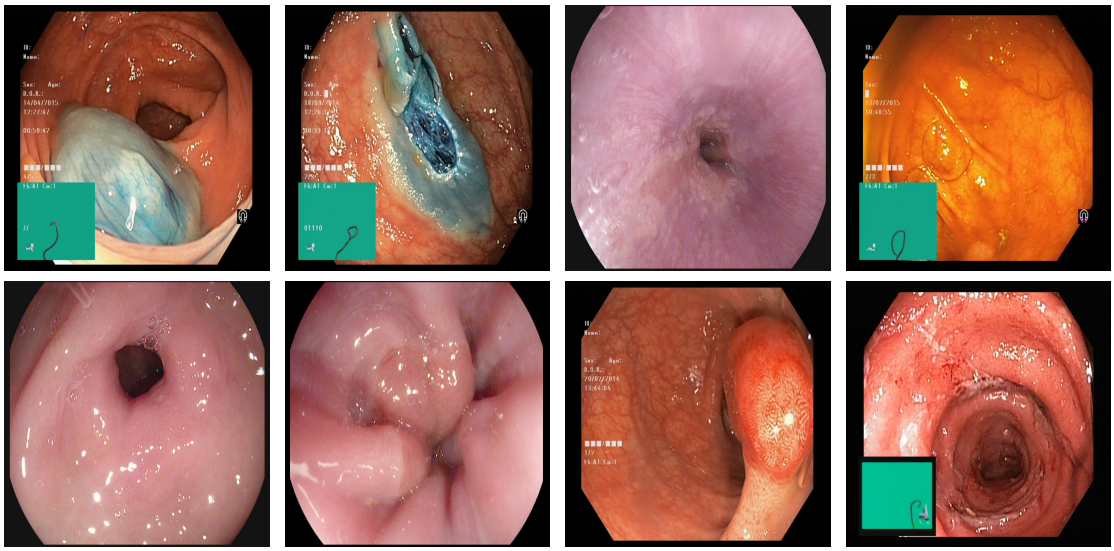


Figure 4.6: Eight example images taken from the Kvasir (v2) dataset [76], one from each class.

2012 is often used to evaluate general purpose convolutional networks as a metric of how well the network performs. The dataset itself contains a vast number of classes, ranging from different species of animals (cats, dogs, horses), to everyday objects (vehicles, furniture, etc.). In Figure 4.5, we see a few sample images taken from eight different classes included in the ImageNet validation dataset.

Keras provides pre-trained convolutional neural networks, trained on images from the ImageNet database, which we will be using as the base weights for our implementation. A table of the various Keras models evaluated against the ImageNet ILSVRC validation dataset can be seen in Table 4.1. The ImageNet validation dataset is often used as an evaluation metric for how well a convolutional neural network performs on everyday objects such as cats, vehicles or furniture to name a few. Using a top-1 and top-5 error score to determine the accuracy of a network. The top-1 error is calculated by taking the highest predicted class and comparing it to the ground truth. The top-5 error is calculated by checking if the target class is part of the five highest predicted classes of the network.

4.1.2.2 Kvasir

Kvasir [76] is a dataset consisting of colored images taken from the GI tract with resolutions ranging from 720×576 to 1920×1072 pixels. The images were collected using endoscopic equipment from Vestre Viken Health Trust (VV) in Norway, with each image being carefully annotated by trained endoscopists. There are currently two versions of the dataset, Kvasir (v1) and Kvasir (v2), consisting of 4000 and 8000 images respectively. Each version of the dataset splits its content equally between 8 classes of either

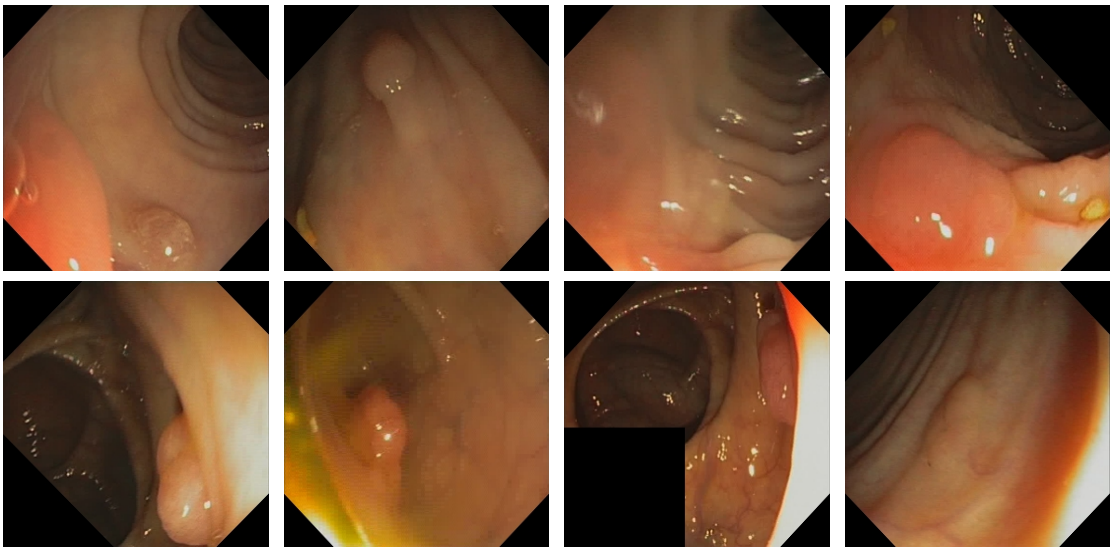


Figure 4.7: Eight example images taken from the CVC-968 dataset. Note that all images are of polyps and have already been pre-processed, with artifacts such as the navigation box removed [16].

anatomical landmarks, pathological findings or endoscopic procedures. Some of the included images (more prevalent in some classes than others) include a green picture illustrating the position and configuration of the endoscope inside the bowel. When visible, it is located in the lower left corner of the image and is used by electromagnetic imaging systems, such as ScopeGuide, Olympus Europe, to extract additional meta-information from the performed endoscopy. This is the main dataset we used for training, meaning it is from this dataset we are trying to learn the features of the eight different classes. As for which version, we used Kvasir (v2) as it contains double the number of images (8000 total). Figure 4.6 shows an example of each of the eight classes of Kvasir.

4.1.2.3 CVC-968

CVC-968 [16] consisting of videos taken from a GI examination. Of the provided videos, we compiled a set of 500 video frames containing signs of polyps. The main use of this dataset was to further evaluate our trained models by introducing images taken from an entirely different dataset. Figure 4.7 shows eight example images taken from different video segments.

The video frames take from CVC-968 differ from the polyp images from the Kvasir (v2) dataset in a variety of ways. Firstly, the video frames from CVC-968 do not contain any artifacts in the form of overlaid text, navigation box (green box located in some images found in Kvasir (v2)), and noise that would otherwise distract from the contents of the image. Secondly, the video frames from CVC-968 are spatially smaller compared to

Level	Category	Name	Version
Hardware	GPU	Nvidia GTX 1080 TI	
	CPU	Intel i7-7700K	
	Memory	Corsair 16Gb DDR4	
Software	Operating System	Ubuntu Xenial Xerus	16.4
	Library	Python	3.6.2
		Tensorflow	1.3.1
		Keras	2.0.8
		CUDA 8	8.0.61
		cuDNN 6	6.0.21

Table 4.2: A table showing the system specifications for the machine used for all trainnig and evaluation sessions.

those of the Kvasir (v2) dataset. Sizes in the CVC-968 vary from 240×240 to 270×270, where images from Kvasir range from 720×570 to 1280×1024. This should not be too much of a problem though, seeing as all images will be re-sized before being sent into the neural network for classification.

4.1.3 Training

In anticipation of training multiple networks, using a variety of CNN architectures and potentially different dataset pre-processing steps, we had to prepare some strategy to make sure each model performance measure was consistent. As we were not too worried about the overall performance of each network (we were not aiming for record numbers), choosing the perfect hyperparameters were not too concerning as long as we got a relatively okay score (hyperparameter selection will be discussed in Section 4.1.3.1). We were, however, anticipating that we would have to train each network multiple times depending on which pre-processing steps we decided to test based on the analysis done on the initial training of each model. To make sure the change in score was due to the pre-processing steps and not other variables, each network would be trained using the same hyperparameters (independently of each architecture of course). As for the training process of each model, we would use the pre-built Keras models together with the pre-trained weights to perform the initial testing of each architecture. As we were going to use Keras' weights trained on the ImageNet dataset, we were going to have to use a technique of re-training trained networks called *transfer learning*.

Transfer learning is the process of reusing pre-trained weights and applying them to a different domain. This is commonly used among pre-built architectures, such as VGG, ResNet, etc., as training these networks can take a considerable amount of time. Additionally, when the amount of data is a constraint, such as in the medical sector, transfer learning can

Model	Epochs	Batch Size	Layers Freezed	Optimizer	Learn Rate
VGG-16	50	32	15	Nadam	0E-6
VGG-19	50	32	17	Nadam	0E-6
ResNet-50	50	32	148	Nadam	0E-6
Inception (v3)	50	32	249	Nadam	0E-6
Xception	50	32	132	Nadam	0E-6

Table 4.3: A table showing the hyperparameters selected for each model.

aid in training the network in basic features for which the smaller dataset can be trained on top. The most basic form of transfer learning is only re-training last block of the network, the block responsible for classification. Let us say that we wanted to train a network to detect ten different types of animals. We could then use a pre-built model for Keras using the weights trained on ImageNet (as ImageNet is already well trained on different types of animals) and apply it to our problem. There is, however, a problem here, the pre-built network is set up to classify 1000 different classes of ImageNet and not the ten classes we want. To use the pre-trained weights, all we have to do is pop off the last block of the network, and replace it with a classification block suitabl£e for our needs (10 classes in this case). Then we freeze the layers leading up to this classification block to withhold the already learned features and train the classification block using the dataset for our ten classes. The result is a model with the knowledge of features obtained from millions of images taken ImageNet, applied to the ten classes for our new problem set. This, however, only works well if the pre-trained problem set is relatively similar to the new domain. In the case of images take from GI endoscopy, where images show little resemblance to those from for example ImageNet, we must perform an additional step where we re-train other parts of the network as well.

This additional step is commonly referred to as *fine-tuning* the model. Under fine-tuning, we re-train different parts of the model in addition to re-training the classification block. This re-training of the network is mostly dependent on how different the original problem is from the new one. As we previously discussed, CNNs learn more abstract features increasingly as layers get deeper. This is the general hypothesis of fine-tuning, as we assume that the initial layers of the network learns features such as basic edges and curves, and therefore do not need to be re-trained. As we get closer to the classification block, learned features get more abstract and become more specific to the problem at hand. It is, therefore, most common to re-train layers moving back from the classification block, where we often set a fixed boundary to which we freeze layers moving backward. Please note that the number of layers frozen under fine-tuning is considered a hyperparameter.

As for our training strategy, we will be using the pre-built Keras models with the included weights trained for validation on the ImageNet dataset, and apply transfer learning by re-training the network on the Kvasir (v2) dataset. Keep in mind that the difference between ImageNet classes and the ones contained in Kvasir (v2) is quite significant, so the rule of transfer learning may not apply as well as if we were training a more traditional network. It is important to note that for the visualizations to work when replacing the classification block of each network, one must use a global pooling layer in place of the more traditional fully connected layers before classification. This was further explained in the Section 2.2.6.2, where we described the production of CAMs.

As we knew that we would be training many networks, sometimes maybe multiple times, we decided to develop a simple way of training and re-producing each network. This system uses JSON based configurations to painlessly train a network without having to make changes in the codebase. The system works by automatically reading configuration files located in a directory, which then automatically trains and evaluates the network based on the parameters set in the JSON file. For fine-tuning, the system allows for linking configuration files so that configurations for base training and fine-tuning are entirely separate. The main advantage of such a system is that it makes it much easier to reproduce previous experiments by wrapping all model configurations into a single file (two in the case of fine-tuning), something that is undoubtedly lacking among deep learning literature. The system in its entirety together with the configuration files used for training can be found on GitHub [43], along with some documentation of how to use it. Additionally, the configuration files for all experiments are visible within the repository as well.

With this system in place, we were ready to select our hyperparameters and start training our models. In the upcoming two sections, we will discuss the process of choosing these hyperparameters and have a brief look at how we kept track of each trained model, making sure not to mix results and model configurations.

4.1.3.1 Hyperparameter Selection

As is quite common in the field of deep learning, the selection of hyperparameters did not follow a strict scientific process. Although we use pre-built architectures, there is still the case of selecting optimization parameters, e.g., learning rate, epochs, batch size, etc. For the most part, these hyperparameters were chosen through a combination theory based on existing literature regarding the various networks, and a tiny bit of trial and error to get somewhat decent performance out of the networks. In addition to the existing literature, there was an influence on the selection based our machine specifications (such as the memory requirements for large batch sizes), and the constraint of working with such a small dataset. As the performance of our models is not the objective of this thesis, we

decided not to spend too much time on tuning the hyperparameters for maximum accuracy. With that said, Table 4.3 shows the hyperparameters selected for each model.

4.1.3.2 Keeping Track of Experiments

As one can image, training multiple networks using different hyperparameters can quickly become messy. To keep track of our experiments, we used the python package "Sacred", which is a tool to help organize, log and reproduce experiments, specially fitted for training neural networks. This allowed us to easily keep track of which networks were trained using which hyperparameters, without accidentally mixing up the performance number of the various models.

4.2 Evaluation Method and Metrics

To evaluate whether or not the data augmentation techniques made any difference in the performance of our models, we trained each network using a method of k-fold cross-validation, 2-fold in our case, resulting in 500 images per class being used for training and validation. In k-fold cross-validation, the dataset is split into k sub-samples (in our case 2), where one sub-sample is used for validation, and the rest are used for training. This process is repeated k times, once for each split in the dataset. After a network has been trained and evaluated k times, we calculate our metrics and average the result. The advantage of this method is maximizing the utility of our dataset, i.e., every image will play a part in the total score, and be part of both training and evaluation. This is especially important when the dataset used is particularly small (which in our case, it is). As discussed in Section 4.1.3, the evaluation phase of our network is part of the training system developed over the course of this thesis. In the upcoming few sections, we will discuss the various evaluation techniques and metrics used to evaluate each model. Additionally, it is important to mention that we did not mix the variations of the Kvasir (v2) dataset, meaning each network is validated on the same dataset used for training (unseen data from the same dataset of course).

4.2.1 Confusion Matrix

Before we look at the specific metrics used to evaluate each model, it is important to understand how these metrics are produced. During the evaluation of our models, we pass each image form the validation split through our network to get a prediction. The highest prediction is the suggested class by the network. This prediction is then plotted into what is commonly referred to as a confusion matrix, or confusion table, which compares what the model predicted against the actual ground truth.

	A	B	C	D	E	F	G	H
A	376	28	1	11	0	1	92	16
B	122	470	1	16	1	0	27	32
C	0	0	424	0	45	186	4	7
D	0	0	0	436	0	0	8	30
E	0	0	0	0	431	5	8	2
F	0	0	73	0	22	308	1	1
G	2	2	1	33	1	0	352	80
H	0	0	0	4	0	0	8	332

Figure 4.8: A confusion matrix taken from the evaluation of one of our model. Please note that the each class has been replaced by the first letters of the alphabet.

Figure 4.8 shows one of the confusion tables produced during one of the evaluation sessions, here we see a table of values, each corresponding to a prediction and the actual label. The vertical and horizontal axis represents each class in our models, denoted by the initial letters of the English alphabet, where the vertical axis is what the model predicted, and the horizontal axis is the ground truth. Note that all presented confusion tables will have each class denoted as alphabetic letters, so for clarity's sake, we have included a legend showing the mapping between each class and letter:

A: Dyed Lifted Polyps

B: Dyed Resection Margins

C: Esophagitis

D: Cecum

E: Pylorus

F: Z-line

G: Polyp

H: Ulcerative Colitis

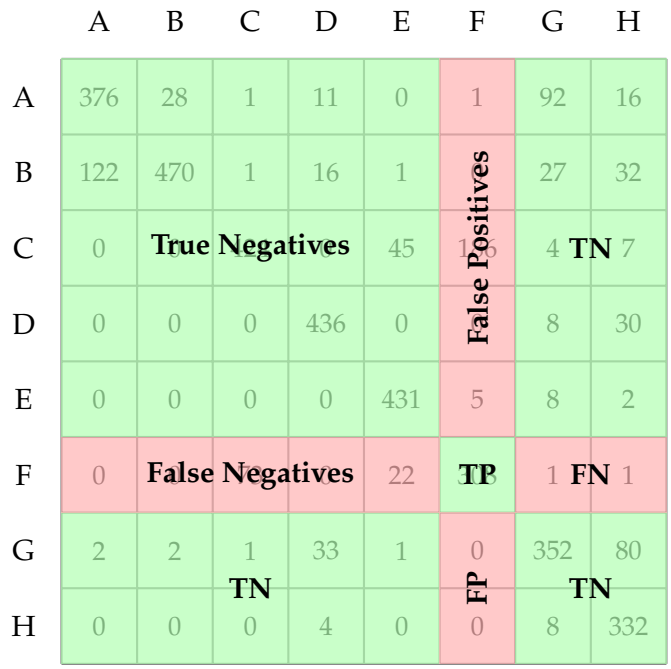


Figure 4.9: An example of how we calculate TPs, TNs, FPs and FNs from a confusion table.

The results is that correct prediction are found in vertical line of cells starting at the top left and ending at the bottom right. Numbers outside this vertical line are incorrect predictions. The confusion table is useful when calculating the true positive (TP), true negative (TN), false positive (FP) and false negative (FN) of a given evaluation. Figure 4.9 gives a good intuition of how we use the confusion table to calculate these metrics.

4.2.2 Metrics

After creating the confusion table, we calculate the TP, TN, FP and FN. These values are calculated using a confusion matrix as shown in Figure 4.9. The Metrics we use for evaluation are as follows; accuract (ACC), precision (PREC), recall (REC), specificity (SPEC), F1-score (F1) and the Matthews correlation coefficient (MCC). Note that each of the described metrics aims to be as close to 1 as possible, meaning lower numbers are not preferred.

The ***Accuracy*** is merely the percentage of correct predictions made by the network (calculated as seen in equation 4.1). This is probably the most common way to measure the performance of the network but does not give a comprehensive overview of how well the models are doing. We therefore often use additional performance metrics to get a more real sense of its competence.

$$ACC = \frac{TP + TN}{TP + FP + FN + TN} \tag{4.1}$$

Precision, sometimes called the positive predictive value, is a measure of relevance, meaning a how many of the positive cases were correctly classified. It is calculated using the formula seen in equation 4.2. This measure is highly correlated with the REC measure, as together they are used to indicate the success of classification when class results are very imbalanced.

$$PREC = \frac{TP}{TP + FP} \tag{4.2}$$

Recall, also known as sensitivity, is a measure of how many relevant cases were found, meaning how many of a particular class were correctly identified as such. It is calculated using the formula seen in equation 4.3. As we mentioned in the description of PREC, PREC and REC are commonly used together and give a more wholesome view of the performance compared to looking at each metrics independently. This is a little difficult to understand but lends itself well to an example. Let us say we have a dataset containing 10000 total cases, where 9990 cases are negative, and 10 cases are positive. If we were to classify all cases as negative, we would have an easy 99.9% accuracy. On the other hand, calculating the REC and PREC for this instance would result in an undefined number (0 in the numerator position) and would give a terrible score. This simple example shows that the accuracy does not show the entire picture, definitively when classes may be imbalanced. However, using two scores, REC and PREC, to score a model is a bit cumbersome when it comes to comparison between different models, it would be convenient if we could compare just a single number. This is where the F1 measure is useful.

$$REC = \frac{TP}{TP + FN} \tag{4.3}$$

The ***F1-score***, sometimes called F-measure, is a weighted average of the precision and recall and is commonly used to directly compare the performance of two (or more) classification models. This measure is frequently weighted dependent on the problem at hand, i.e., PREC and rec are commonly weighted differently. Areas where incorrect classifications pose a significant risk, such as diagnosing cancer patients, PREC should be more heavily weighted than REC. Same goes the other way. If misclassification is not that important, such as incorrectly classifying two species of cats, one can weight REC higher than PREC. In our case, even though the target classificatio£n is on medical imaging data for diagnosis of severe disease, for our current experiments, misclassifications are not too alarming, so for all models, we will be weighting REC and PREC equally.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{4.4}$$

Specificity, also called the negative rate, measures the number of negative cases correctly classified as such. This measure is closely related to REC, and often referenced together as specificity and sensitivity (REC). In the same way, REC and PREC gave a more wholesome view of the performance of the model, so does specificity and sensitivity. Increasing specificity, i.e., increasing the likelihood that a given case is true, decreases sensitivity, while increasing sensitivity, i.e., increasing the probability of a positive result, decreases specificity. One sees how these two metrics compliment each other and should be weighted in the same way as PREC and REC.

$$SPEC = \frac{TN}{TN + FP} \tag{4.5}$$

The last metric we will be using is the ***Matthews correlation coefficient***, which is a measure that takes all TP, TP, TP and TP into account. Similarly to the F1, the MCC gives us a better indication of the total performance of a model compared to looking at the previous metrics individually. The MCC differs from the other metrics in that its value can be between -1 and 1 depending on the distribution of data.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{4.6}$$

By taking the average F1-score of each split of our models, we can directly compare what models perform better or worse depending on their F1-score. Note that comparisons are between the same type of model, e.g., we will not make comparisons between an Xception and a VGG-19 network as we are not necessarily testing performance between networks, but testing the quality of the dataset used to train them. In addition to performing 2-k cross-validation, we will be re-evaluating each model using images from a different dataset to test the generalizability of each network further. To do this, we will be adding 400 polyp images taken from the CVC dataset [16].

4.2.3 Model Evaluation

There are multiple ways of evaluating a model. Firstly, the most common way to measure the performance of a model is through simple classification and verification, e.g., classify an image and log the results using the beforementioned metrics. Secondly, one may measure the performance of a model by looking at what the model "sees". This mode of evaluation is more extensive and is common among models trained for image segmentation, locating objects and boundaries within an image. Both are

essential methods, but require different types of labeled data to perform. In the next couple of sections, we describe our evaluation process, which solely relies on evaluation the classification ability of each model. Sadly, we did not have the required labeled image data to perform evaluations on the produced visualizations but have included a section on the topic.

4.2.3.1 Evaluation of Classification

Using 2-fold cross-validation, in addition to evaluating each architecture using both the base set of Kvasir and Kvasir with an added 500 polyps taken from CVC-968, each architecture is evaluated a total of four times. After training a model split, we first evaluated using the standard validation set of Kvasir. This is done by loading the highest performing weights produced during the training session and running each image of the validation set through the model. The highest predicted class is then compared against the ground truth and logged in the confusion matrix (seen in Figure 4.8). After having run through all the images, we calculate the TP, TN, FP and FN using the confusion matrix in the manner shown in Figure 4.9. These metrics are then used to calculate the evaluation metrics described in Section 4.2.2. With the produced metrics for one split of the model produced, we repeat this process for the second split. With both splits evaluated, we average the scores to get the final metrics for the trained architecture. This same process is done when evaluating using the additional Polyp images, the only thing that changes is that we move the 500 images to the validation set before evaluating. This described process is repeated for each of the trained architectures.

4.2.3.2 Evaluation of Localizations

With the various evaluation metrics explained, we would also like to include the process for evaluating image localization techniques, even though we were not able to perform this evaluation as we lack the necessary ground truth maksing of Kvasir (v2). This method of evaluating each model gives a true sense of the performance of a CNN, as it shows how many images were correctly classified based on the localization of activations for the given image. This means that even though a model might correctly classify an image, it might not necessarily detect the object in question. Evaluation using this method requires more than just annotated images, but also requires a bounding box to validate the object in question. With a CAM and bounding box pair, we can verify if the classification was correct or not by how much of the activations fall within this box, thus correctly verifying that the image is collected. This measure would be beneficial for this work, but we sadly lack the necessary data to perform these experiments. We will therefore only be looking at the general evaluation metrics.

Evaluated on Kvasir (v2)

Model	PREC	REC	SPEC	ACC	MCC	F1
VGG-16	0.977	0.841	0.813	0.959	0.834	0.833
VGG-19	0.975	0.830	0.799	0.955	0.822	0.822
Inception (v3)	0.975	0.830	0.802	0.956	0.825	0.825
Xception	0.980	0.859	0.838	0.964	0.858	0.858
ResNet-50	0.980	0.864	0.841	0.965	0.860	0.860

Table 4.4: The evaluation results of all models trained on the "vanilla" version of Kvasir (v2).

Evaluated on Kvasir (v2) + CVC-968

Model	PREC	REC	SPEC	ACC	MCC	F1
VGG-16	0.966	0.770	0.743	0.940	0.791	0.768
VGG-19	0.966	0.771	0.739	0.940	0.784	0.766
Inception (v3)	0.971	0.803	0.774	0.949	0.806	0.801
Xception	0.976	0.836	0.816	0.959	0.843	0.838
ResNet-50	0.975	0.826	0.805	0.956	0.837	0.827

Table 4.5: The evaluation results of all models trained on the "vanilla" version of Kvasir (v2), with added polyps from CVC-968 used in evaluation.

4.3 Initial Training Results

Tables 4.4 and 4.5 show the results of our initial training and evaluation of the described architecture trained on the Kvasir (v2) dataset, using the hyperparameters detailed in Table 4.3. Looking at the metrics, we see that each model performs somewhat well, perhaps not well enough for medical usage, but well enough for us to conduct our experiments. Notice that, for the most part, all metric scores drop when introducing the polyps from the CVC-968 dataset. This indicates that the models do not generalize well to completely new data, i.e., data that is different from what was seen under training. Of the bunch, we see that the model based on Xception retains most of its score, with the VGG-16 model falling the most. It is worth noting the images contained within the CVC-968 dataset have gone through some pre-processing, and are clear of any artifacts commonly found in Kvasir, e.g., green navigation box, text, etc. This may be one of the reasons why our models do not work too well on these images, as they may have learned to associate some of these artifacts with the class of "polyp", which is the only additional class images added by CVC-968. Note that we only add 500 additional polyp images, accounting for approximately 11% of the overall validation set, the drop in metrics is relatively high (almost 7 points for

	A	B	C	D	E	F	G	H
A	865	151	0	8	2	1	51	6
B	111	841	0	0	0	0	7	6
C	0	0	829	0	32	380	6	5
D	0	1	0	795	0	0	12	10
E	0	0	7	0	919	14	11	5
F	0	0	157	0	27	600	5	5
G	18	5	2	113	11	4	805	46
H	6	2	5	84	9	1	103	919

(a) VGG-19 Confusion Matrix

	A	B	C	D	E	F	G	H
A	812	248	0	2	1	1	28	6
B	122	716	0	2	0	1	4	5
C	0	0	790	0	35	238	3	4
D	6	4	0	889	1	0	44	29
E	0	0	3	0	898	6	7	4
F	0	0	202	0	39	751	3	0
G	47	16	1	58	16	3	851	59
H	13	16	4	49	10	2	60	893

(b) Inception (v3) Confusion Matrix

Figure 4.10: The confusion matrices produced under the initial evaluation of our trained VGG-19 (4.10a) and Inception (v3) (4.10b) based models.

	A	B	C	D	E	F	G	H
A	889	199	1	5	2	3	26	2
B	98	794	3	0	0	0	0	0
C	0	0	730	0	4	202	0	1
D	0	2	0	916	1	0	42	28
E	1	0	2	0	964	2	5	3
F	1	0	260	0	4	787	1	0
G	9	3	0	46	24	5	878	29
H	3	2	4	33	1	1	48	937

(a) ResNet-50 Confusion Matrix

	A	B	C	D	E	F	G	H
A	829	191	0	0	1	0	16	2
B	120	789	1	0	0	0	0	4
C	0	0	802	0	16	245	0	1
D	5	6	0	946	1	0	34	40
E	0	0	8	0	949	5	11	4
F	0	0	185	0	24	748	0	0
G	40	6	1	38	7	2	904	56
H	6	8	3	16	2	0	35	893

(b) Xception Confusion Matrix

Figure 4.11: The confusion matrices produced under the initial evaluation of our trained ResNet-50 (4.11a) and Xception (4.11b) based models.

VGG-16). However, looking at the metrics alone does not give us the full picture. Looking to the confusion matrices might give us a better indication of where our models are failing.

Figure 4.10 and Figure 4.11 show the calculated confusion matrices for each model (combined across both validation splits and excluding the

added polyps). Looking at the confusion matrices, we see that some classes are generally confused with others, and this usually goes both ways, i.e., if class A gets confused with class B, class B typically gets confused with class A. Looking at which classes get confused is a good starting point for discovering potential training improvements. It is important to mention that some classes within the Kvasir dataset look inherently similar to each other, but there are also non-natural similarities such as an imbalance of different artifact types which may be the cause for the confusion. With this in mind, we looked at the confusion matrices to see what classes we should analyze further.

What we found was a collection of five class pairs, each commonly mistaken by each of the trained models. The class pairs were as follows (note that we have denoted each class with a letter from the English alphabet, denoting their label in the confusion matrix);"dyed lifted polyps" (A) and "dyed resection margins" (B), "esophagitis" (C) and "z-line" (F), "cecum" (D) and "polyp" (G), "cecum" (D) and "ulcerative colitis" (H), and "polyp" (G) and "ulcerative colitis" (H). Some of these confusions may be well founded, e.g., classes that look very similar to each other or contain traces of multiple classes, but this needs further analysis before concluding.

4.4 Analysis of Initial Training Results

Based on our findings in the previous section, we decided to conduct further analysis on the confused class pairs with the purpose of finding the cause of this confusion. This analysis was primarily done using Mimir's neural network dissection tool, for which we looked at what areas of the confused image activated for the confused class. But, before starting the analysis, we first had to do some preparation work, i.e., set up a strategy of how we were going to analyze each class pair using each trained model. The reason behind setting up a plan beforehand was to avoid blending results, leading to incorrect model assumptions. This might not have been too useful in the first round of analysis where we only have to keep track of five models, but came in handy when we later had to keep track of the analysis coming from 15 different models. The analysis process can be split into four separate stages, and are performed as follows:

1. Start by selecting one model to analyze and look up the corresponding confusion table to see what classes were commonly confused with each other.

2. With a class pair selected, upload images of the confused class to the neural network dissection tool through the manual upload button located in the upper right corner of the UI.

3. Select an image where the model gets confused with the other class and use the visualization tool to look at what regions activate for both the confused class and the confused image.

4. Clear the uploaded images and repeat for all models and class pairs.

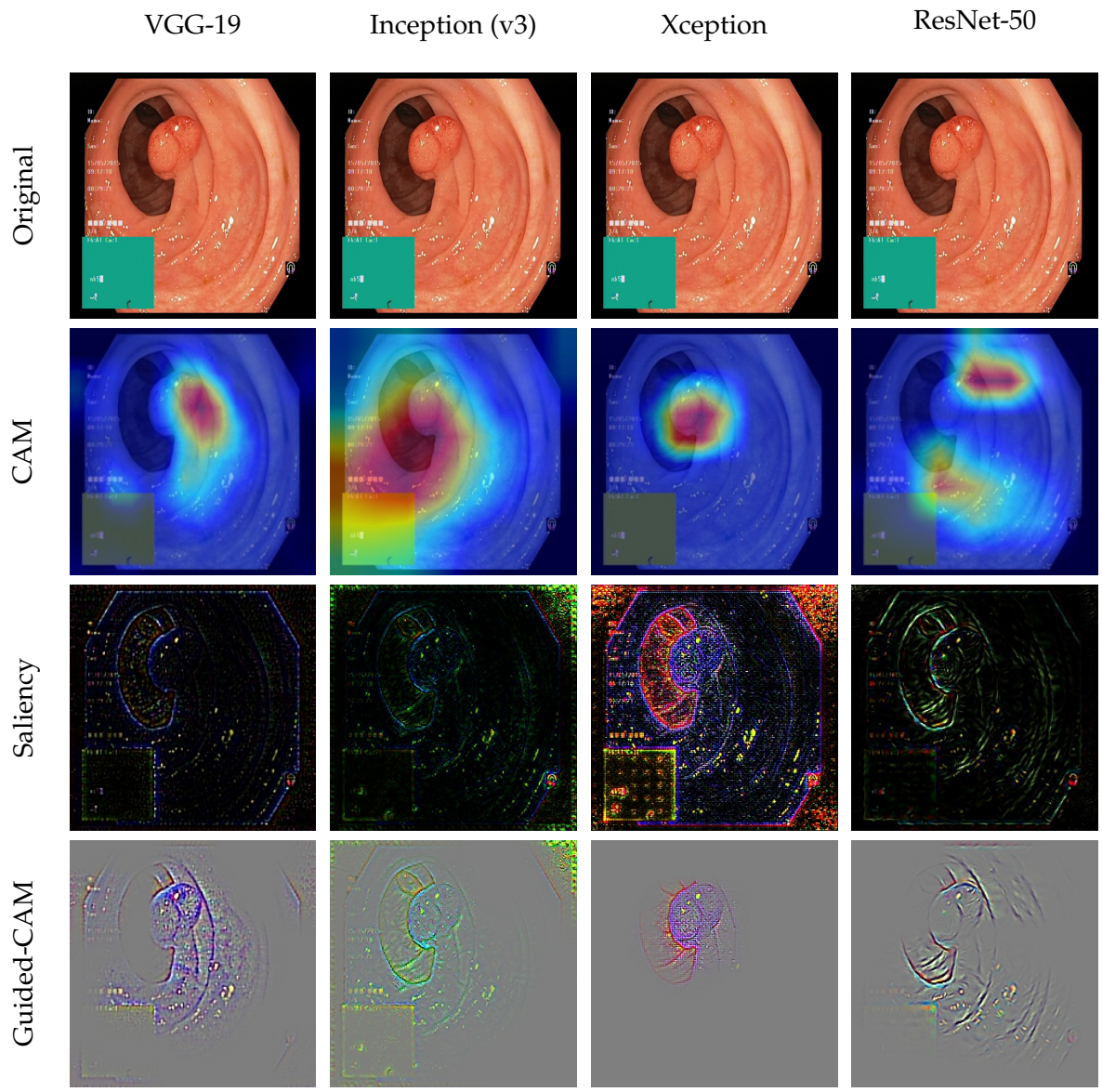


Figure 4.12: A table of images showing the different visualizations across different architectures, using an image containing a polyp. Each visualizations is done using the last convolutional layer of the network and has the *polyp* class as target. Note that the "last" convolutional layer in an architercure may be a bit missleading as some networks (suchs inception (v3)) have parallel convolutional layers. To keep things simple, we decided to go with the last convolutional layer as they are ordererd in Keras.

But, before performing analysis on confused class pairs, we decided to have a look at a case where each model correctly identified and localized an image. This is interesting as it shows, at the very least, that our models have some knowledge regarding the features which belong to a given class. Additionally, it is a nice way showcase how each architecture "sees" a

given image at the point of classification. Figure 4.12 shows the selected image, visualized for each of the trained architectures using "polyp" as the target class. Looking at the visualizations, we see that each model correctly localized the polyp, with the Xception architecture seemingly performing the best out of the four models (which in retrospect, makes sense as it was the best performing model when introducing the additional polyp images as shown in Table 4.5).

With at least some knowledge regarding the correctness of our models, we move on to further analyze each confused class pair, which we discuss in further detail in the upcoming few sections. For each of the confused class pairs, we present a collection of three image sets, each set being classified and visualized by a different model. Each image set contains one image, visualized using both the correct class and confused class as targets. Each section will contain a brief discussion regarding each image set, where we look at potential reasons for why each model got confused. With a basic strategy described above, we were ready to started analyzing each model, one by one. Note that due to the amount of data collected during this analysis, we opted only to include the visualizations of models based on VGG-19, ResNet50 and Xception architecture. In the upcomming few section, we give a brief summary of the analysis done on each of the five confused class pairs discussed in Section 4.3. Each section will provide a figure as basis for discussion, which show a comparison between how each network "sees" a given class when making a prediction. This analysis was done using Mimir's neural network dissection tool, which allows us to inspect the final convolutional layer of a given CNN to directly see what correlates to the produced confidence (further explained in Section 3.2).

4.4.1 Comparing Dyed Resection Margin to Dyed Lifted Polyp

First, we looked at cases where the class "dyed resection margin" (A) was misclassified as "dyed lifted polyp" (B), which based on the confusion matrices was common among all models (as seen in Figures 4.10 and 4.11). We already had suspicions in regards to the similarity between these two classes, as they both relate to polypectomy and have a stark blue coloring associated with each of them. Despite their similarities, each class should be visually distinct enough to easily be determined which is which. Figure 4.13 shows the three selected images used for further analysis.

VGG-19 This model barley misclassified the model, missing the correct class by approximately 3%. Looking for the reason behind this misclassification, we start by looking at what the network "sees" when predicting the correct class, "dyed resection margin" (Figure 4.13a). Here we see that the network is correctly able to identify the resection mark, located near the center of the image. This shows that the network has at least learned the specific features of a resection mark, but still manages to classify it as a "dyed lifted polyp". Looking at the confused class visualizations (Fig-

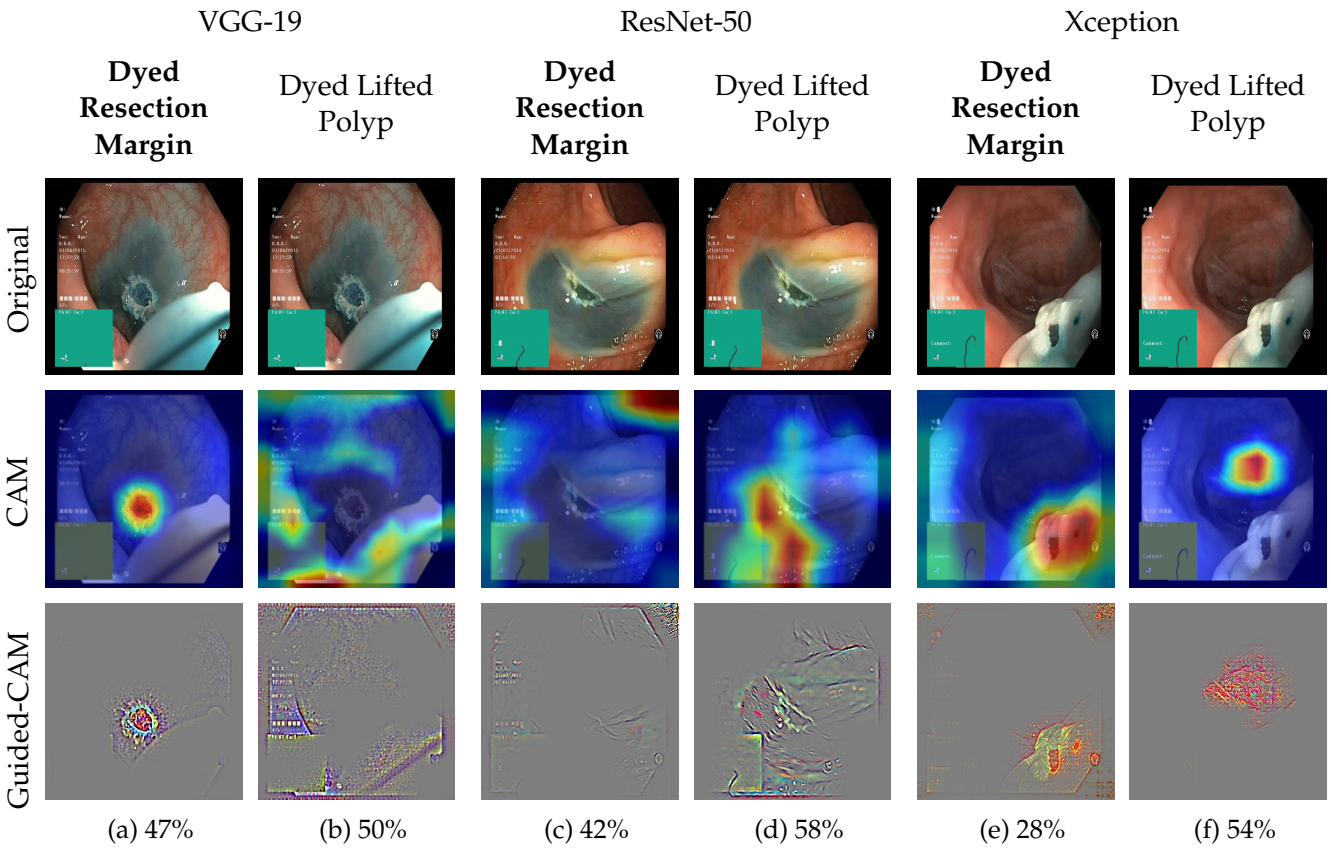


Figure 4.13: Three visualization sets taken from a VGG-19, ResNet-50 and Xception based network, taken from cases where the respective model misclassified the image as "dyed lifted polyp" when the actual class is "dyed resection margin". By looking at the visualization sets, we set out to determine what regions of the image in question led to the misclassification. Each visualization set shows what areas of the given image directly contributes to the predicted score.

ure 4.13b), we see that the network mostly activates on the bottom and top border, in addition to the text and partly on the green navigation box. This is incorrect, as we do not want the network thinking that these artifacts (such as those previously mentioned) are features of a given class. Perhaps the removal of these artifacts would lead to a lower score for "dyed lifted polyps", and thereby correctly predict this image as "dyed resection margin". We note our findings and move on to the next model.

ResNet-50 Similar to that of the VGG-19 model, we see that the network has given both the correct and confused class a relatively high classification score (42% and 58% respectively). Looking at the visualizations for the correct class (Figure 4.13c), we see that the network completely misses the resection mark located near the center of the image. Instead, the network mostly activates on the upper right corner (part of the black border), and

partly on the text and navigation box. Looking at the visualizations for the incorrect class (Figure 4.13c), we see a more sensible activation pattern, with the network mostly activating on the blue dye next to the resection mark. This part is correct, as the network is correctly focusing on the features part of the confuse class ("dyed lifted polyp") and therefore we conclude that the artifacts present in the image distract the network from learning the true features of the correct class.

Xception Similar to the case of VGG-19, the Xception based network is successfully able to localize the resection mark as seen in Figure 4.13e, although its score is much lower. Unlike the VGG-19 model, the Xception model has slight activations on the text located on the right side of the image. This may be partly to blame for this lower score. Looking at the incorrect visualization (Figure 4.13f), we see that no part of the predicted score is attributed to the presence of artifacts. Instead, the network focuses on the back mucosal wall. This is a bit strange, as one would think that the network would instead focus on the blue mucosal, rather than the plain back wall. We conclude that the text is partly to blame for the misclassified image and move on.

In conclusion, we see that the misclassification of each image was due to, in some part, to the presence of text, navigation box, and borders. We marked these artifacts as potential candidates for removal and moved on to the next class pair which was comparing esophagitis to z-line.

4.4.2 Comparing Esophagitis to Z-line

Second, we looked at cases where the class "esophagitis" (C) was misclassified as "z-line" (F) (as seen in the Figures 4.10 and 4.11). Again, we suspected that this class would have some overlap, as esophagitis is a commonly found disease around the z-line [53]. This, however, should not excuse the model from distinguishing a healthy z-line form that of one afflicted with esophagitis. As in the previous comparison, we look at each visualization pair of each model, starting with VGG-19.

VGG-19 Looking at visualizations for the correct class (Figure 4.14a), we see that, for the most part, the network can correctly localize the mucosal breaks located close to the center of the image. This is good, as it has learned to associate this feature with the disease esophagitis. Looking at the visualizations for the class "z-line" (Figure 4.14b), we see that the network activations are quite sporadic. For the most part, these activations are centered around the border edges between the main image and black borders. This can be seen looking at the guided grad-CAM representation of Figure 4.14b. Perhaps removal of these borders would lessen the score for this class, thus resulting in a correct classification. We noted our findings and moved on to the ResNet-50 based model.

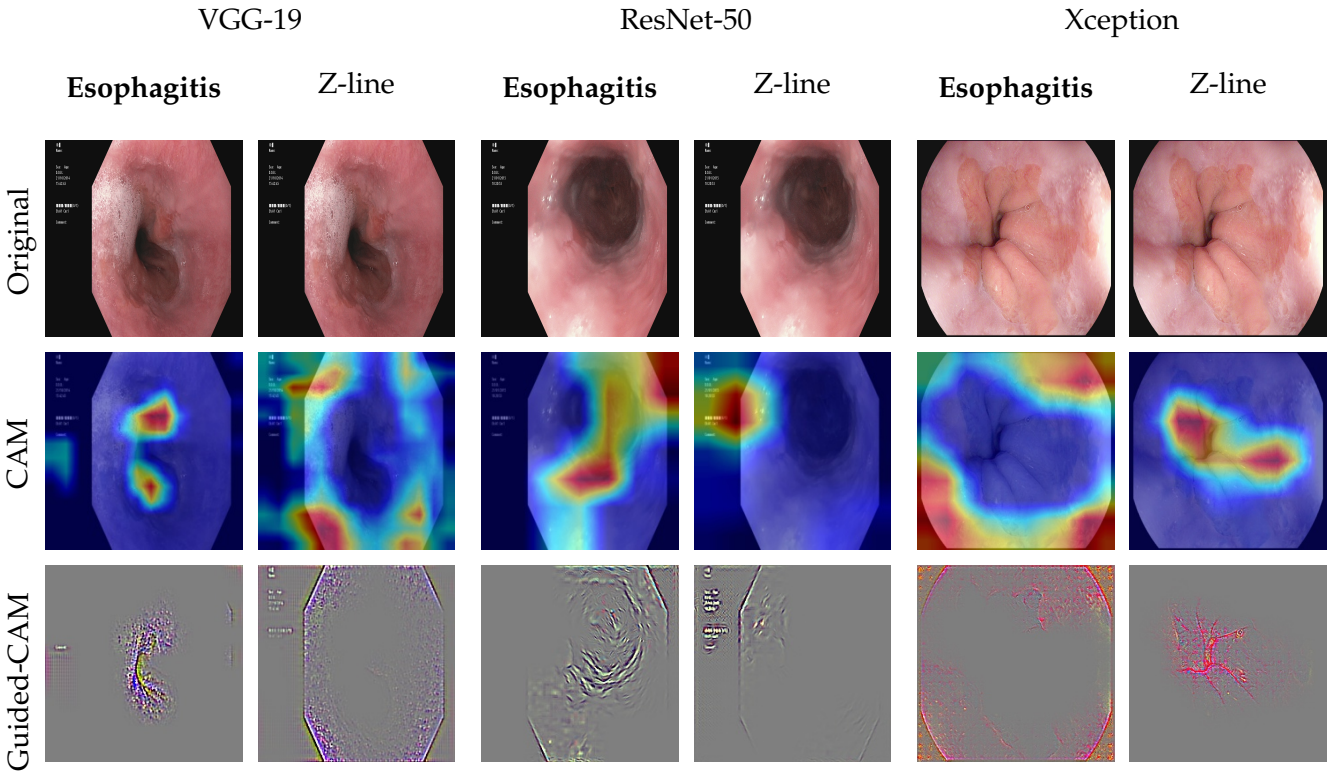


Figure 4.14: Three visualization sets taken from a VGG-19, ResNet-50 and Xception based network, taken from cases where the respective model misclassified the image as "esophagitis" when the actual class is "z-line". By looking at the visualization sets, we set out to determine what regions of the image in question led to the misclassification. Each visualization set shows what areas of the given image directly contributes to the predicted score.

ResNet-50 In a similar case to that of the VGG-19 model, the ResNet-50 based visualizations show that the network correctly identifies features of "esophagitis" when targeting this class (Figure 4.14c). However, in addition to activating on right areas of the image, it also activates heavily on the upper right corner (similar to how the ResNet-50 also focused on the upper right corner when classifying a dyed resection margin). For the visualizations of the incorrect class (Figure 4.14d), we see a similar case to the last, but much more focused on the border and text located in the left region of the image.

Xception Finally, we look at an image misclassified by our Xception based network. This image is a bit different from the previous two, as it shows the z-line afflicted with esophagitis (as suspected at the beginning of this section). First, looking at the visualizations for the correct class (Figure 4.14e), we see that the network mostly focuses on each of the black

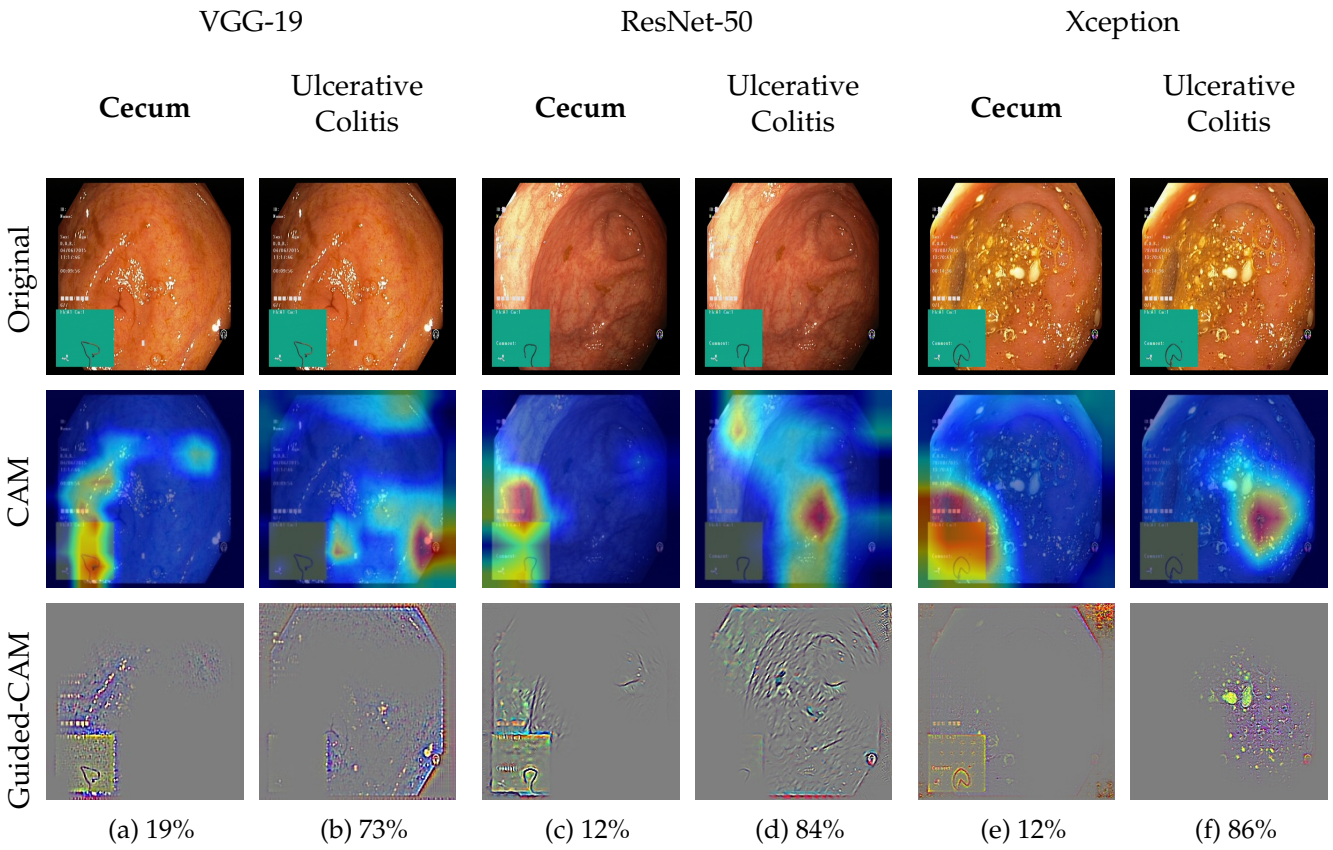


Figure 4.15: Three visualization sets taken from a VGG-19, ResNet-50 and Xception based network, taken from cases where the respective model misclassified the image as "cecum" when the actual class is "ulcerative colitis". By looking at the visualization sets, we set out to determine what regions of the image in question led to the misclassification. Each visualization set shows what areas of the given image directly contributes to the predicted score.

corners of the image, mostly missing the signs of esophagitis located in the central left part of the image. The incorrect visualizations, however (Figure 4.14e), show that the network correctly identifies the parts of the image related to the z-line, specifically the transition from esophageal to gastric mucosa. This is good, but the black border activations of the correct class seem to be getting in the way of the true features of the image. Again, these black borders look to be playing a significant role in the misclassification of many of these images.

In conclusion, we see that the border and text play a large role in the classification of these images, making them good candidates for removal.

4.4.3 Comparing Cecum to Ulcerative Colitis

Third, we looked cases where "ulcerative colitis" (H) was incorrectly classified as "cecum" (C) (as seen in the Figures 4.10 and 4.11). Similar to the last class pair, signs of both classes may present in the same image, as ulcerative colitis may affect the cecum. However, as the class "cecum" denotes a healthy cecum, each network should be able to distinguish between the two classes. We begin with looking at at the image misclassified by our trained VGG-19 model (as seen in Figure 4.15).

VGG-19 Looking at the visualizations using "cecum" as target class 4.15a, we see that the network mostly focuses on the green navigation box and moves up towards the text. Similar to some of our previous cases, the network has incorrectly learned to associate the green navigation box and text the class "cecum". However, this leads to a relatively low score. Looking at the visualizations using "ulcerative colitis" as target class 4.15b, we see that the network the network again mostly focuses on incorrect parts of the image. In this case, it heavily focuses on the anchor object located on the border in the lower right corner. It is strange that neither of the highest predicted classes focused solely on features of each class, perhaps removing the text, navigation box, and anchor would allow the network to learn the true features of each class. We noted our findings and moved on to look at the visualizations generated using the ResNet-50 based model.

ResNet-50 Similar to the visualizations of the VGG-19 based model, visualizing the network using "cecum" as the target class we see that it heavily activates on the navigation box and text (Figure 4.15c). It seems like these artifacts are distracting the network form the true features of the class. Visualizations for the "ulcerative colitis" class shows that the network has learned to associate features of the cecum with this disease (Figure 4.15d). This is partly expected, as ulcerative colitis can often be found in the cecum. However, these features should be more indicative of the "cecum" class, and not "ulcerative colitis". Again, we see that the network gets confused by the presence of various artifacts.

Xception For the Xception based model, we see a very similar case to that of the ResNet based model. Figure 4.15e shows that when visualizing for the correct class ("cecum"), the network mostly focuses on the navigation box. Looking at Figure 4.15f, we again see that the network associates features of cecum with the "ulcerative colitis" class. With all classes seemingly associating the navigation box as a true feature the "cecum" class, we decided to look if there was some disparity between the two classes when it comes to the inclusion of this navigation box. What we found was that the navigation box was part of almost every image included within the "cecum" class, with the "ulcerative colitis" class containing much fewer instances of this artifact. As this artifact is also part of many

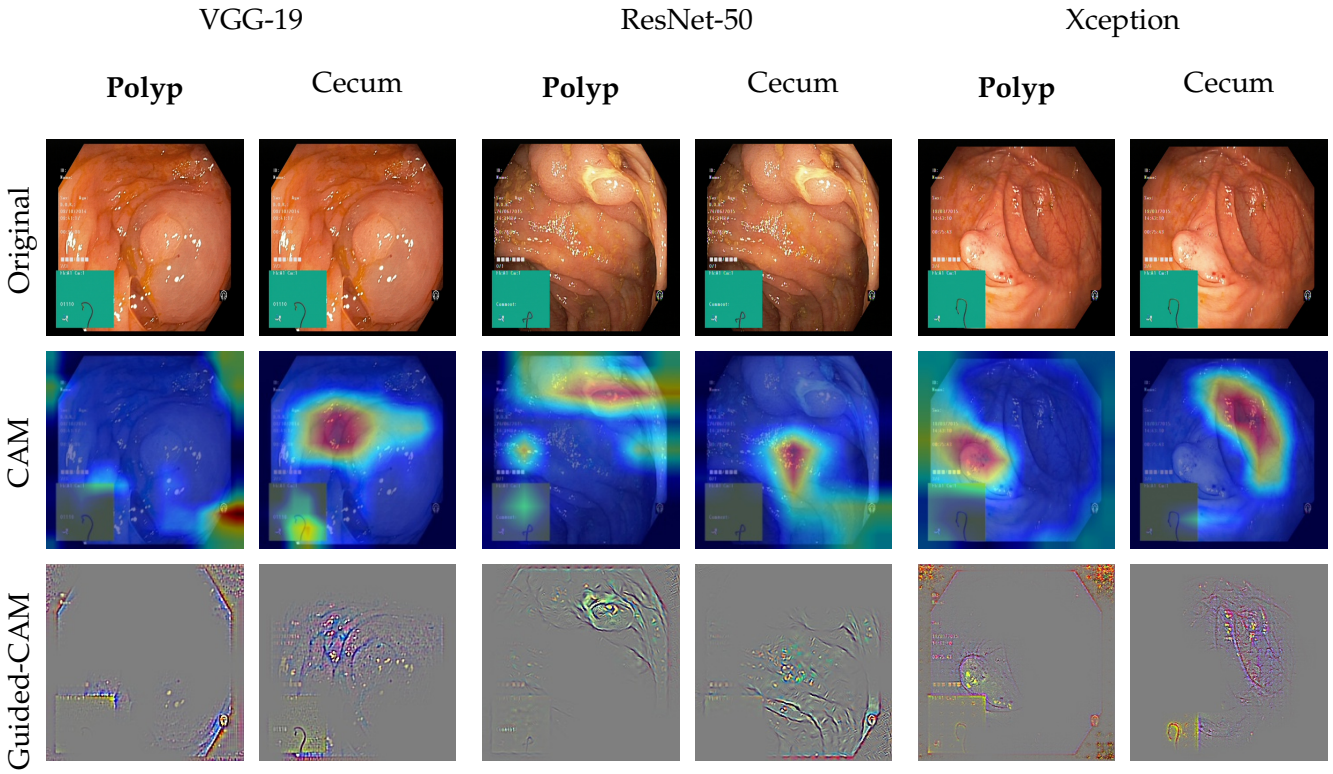


Figure 4.16: Three visualization sets taken from a VGG-19, ResNet-50 and Xception based network, taken from cases where the respective model misclassified the image as "polyp" when the actual class is "cecum". By looking at the visualization sets, we set out to determine what regions of the image in question led to the misclassification. Each visualization set shows what areas of the given image directly contributes to the predicted score.

other images, it seems like this feature is considered "lesser", thus resulting in the relatively lower score.

In conclusion, the navigation box seems to play a significant role in the classification of some images. And that the removal of this box could lead to better classification performance.

4.4.4 Comparing Polyp to Cecum

Fourth, we looked at cases where "polyp" (G) was mislabeled as the "cecum" (C) class (as seen in the Figures 4.10 and 4.11). We again expected some overlap here, as polyps do appear in the cecum. But as polyps are quite visually distinct, we do expect that each network should prioritize a "polyp" classification over "cecum". We start looking at the visualizations for the VGG-19 based model (As seen in Figure 4.16).

VGG-19 Looking at the activations related to the correct class "polyp" for the VGG-19 based model (4.16a), we see that the network focuses almost exclusively on the artifacts present within the image. Mostly focusing on the lower right corner near the anchor like object. Visualizations for the incorrect class, "cecum", show that the network activates on the mucosal wall, with some activations being targeted at the navigation box (Figure 4.16b). This is mostly correct, as the network correctly identifies the cecum part of the image, but is not able to see the polyp. Removing the corners might shift the focus of the network to look at the polyp, potentially leading to a correct classification.

ResNet-50 For the ResNet-50 based model, we see that it correctly identifies the polyp located in the upper right corner (Figure 4.16c). It is therefore strange that the network gives the "polyp" class such a low score. We see in the visualizations for the "cecum" target class that the network mostly focuses on the clear part of the mucosa wall (Figure 4.16d), with lighter activations in the lower right corner. It is difficult to say why this misclassification happens, but perhaps lighter activations are distracting the network, thus resulting in the lower score.

Xception Similar to the ResNet-50 based model, the Xception model correctly detects the polyp located in the images left region (Figure 4.16e), but still gives the class "polyp" a relatively low score. It seems like the Xception network is more distracted by the text and borders then that of the ResNet-50 model, but the main activations still lie on the polyp. Looking at the incorrect class, we see that the network correctly identifies the part of the mucosal wall of the cecum (Figure 4.16f). We again see that the network does not emphasize the detection of the polyp, and therefore mispredicts the class.

In conclusion, not all visualizations are clear when it comes to explaining the exact faults of the classification. Sometimes the incorrect classifications are looking at the correct regions of the misclassified image, but incorrectly emphasizing what is important, i.e., not prioritizing a disease detection. This may be improved by the removal of artifacts, which we saw some light activity, but this may not entirely solve this issue.

4.4.5 Comparing Ulcerative Colitis to Polyp

Lastly, we looked at was comparing cases where "ulcerative colitis" (H) was misclassified as "polyp" (G) (as seen in the Figures 4.10 and 4.11). We begin by looking at the visualizations produced by the VGG-19 based network.

VGG-19 Figure 4.17a shows the VGG-19 network visualized for the "ulcerative colitis" class. We see here that network can identify the white

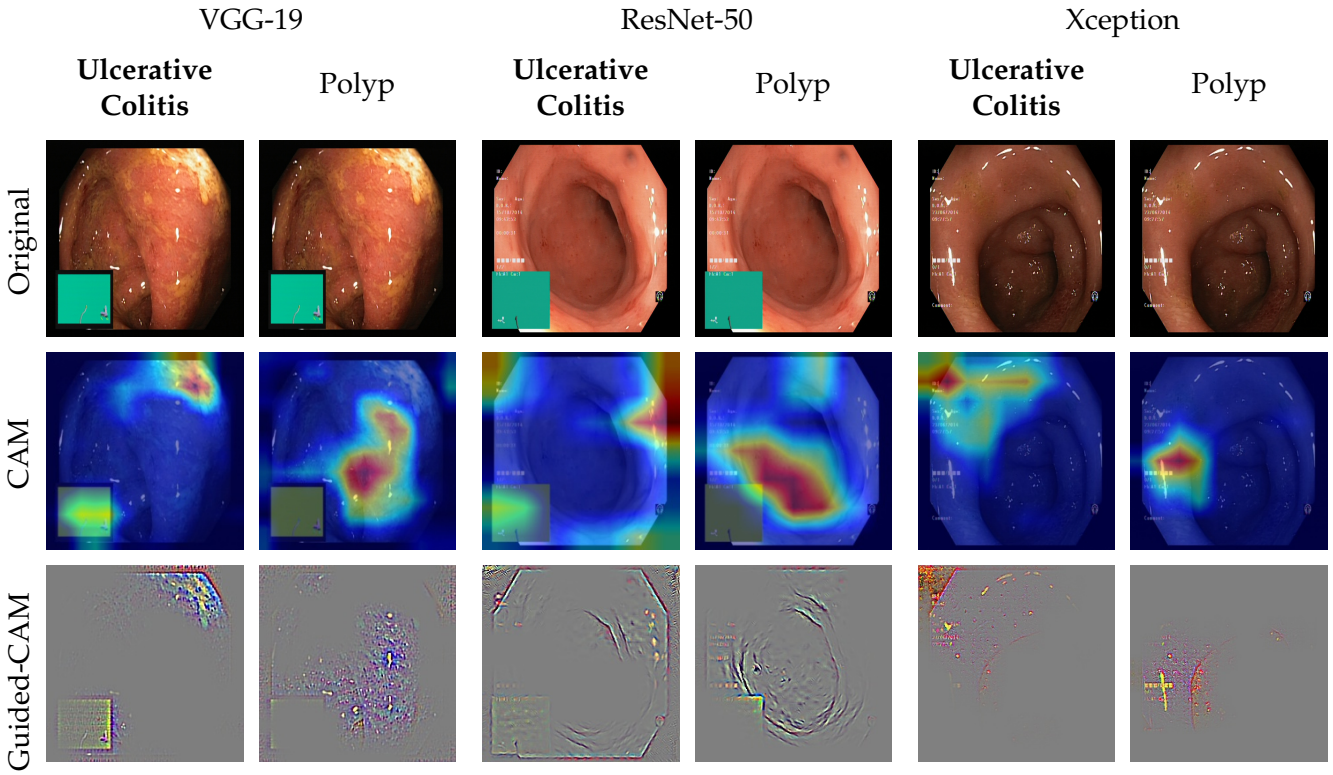


Figure 4.17: Three visualization sets taken from a VGG-19, ResNet-50 and Xception based network, taken from cases where the respective model misclassified the image as "ulcerative colitis" when the actual class is "polyp". By looking at the visualization sets, we set out to determine what regions of the image in question led to the misclassification. Each visualization set shows what areas of the given image directly contributes to the predicted score.

ulcers located in the upper right region of the image. There are some slight activations on the navigation box, but these seem to be quite low in comparison. The visualizations for the "polyp" (Figure 4.17b) class show that the network confuses the transitional depth of the edge (located in the middle of the image) for raised edges of a polyp. This is, of course, incorrect, but it is difficult to determine what one could do to remedy this fault. Based on the two visualizations, we could remove the navigation box with the hope that the network prioritizes the white ulcers seen in Figure 4.17a.

ResNet-50 Moving on to the ResNet-50 based model, we see the visualizations for the correct class heavily activates on the right border of the image (Figure 4.17c). This is yet another case where the border seems to confuse the network. Similar to the last case, the visualizations for the incorrect

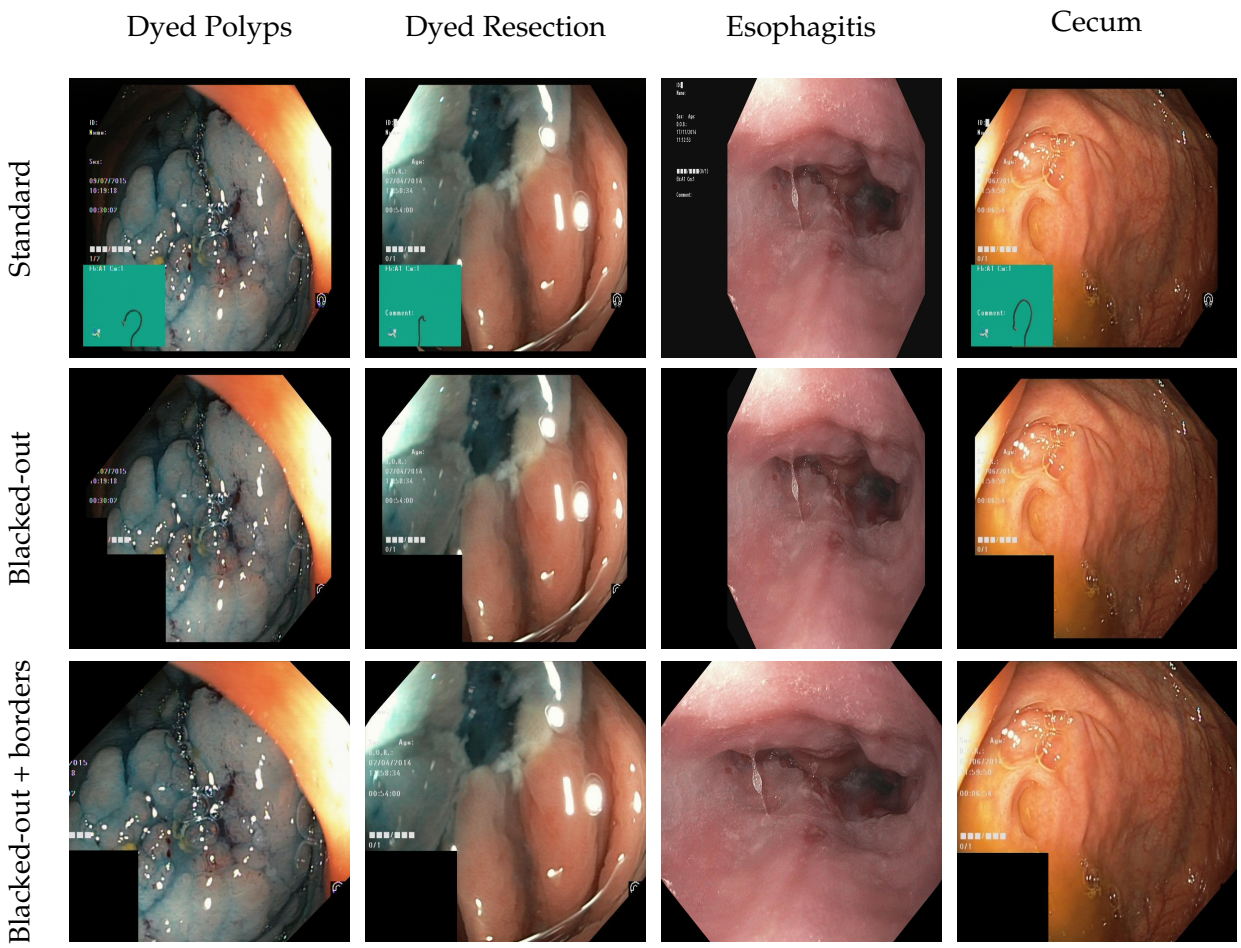


Figure 4.18: Four example images taken from the classes "dyed lifted polyp", "dyed resection margin", "esophagitis" and "cecum", showing Kvasir (v2) after pre-processing.

class seem to indicate that the network confuses the edge depth with the raised edge of a polyp (Figure 4.17d).

Xception In a similar case to that of the last, the Xception based network seems to be activating heavily on the border of the image, distracting the network from the true features (Figure 4.17e). When visualizing the network for the class "polyp", we again see that the network detects the transitional depth edge in addition to parts of the text (Figure 4.17f).

In conclusion, we again see that the various networks associated different image borders with specific classes. This should be taken into consideration when pre-processing data before training.

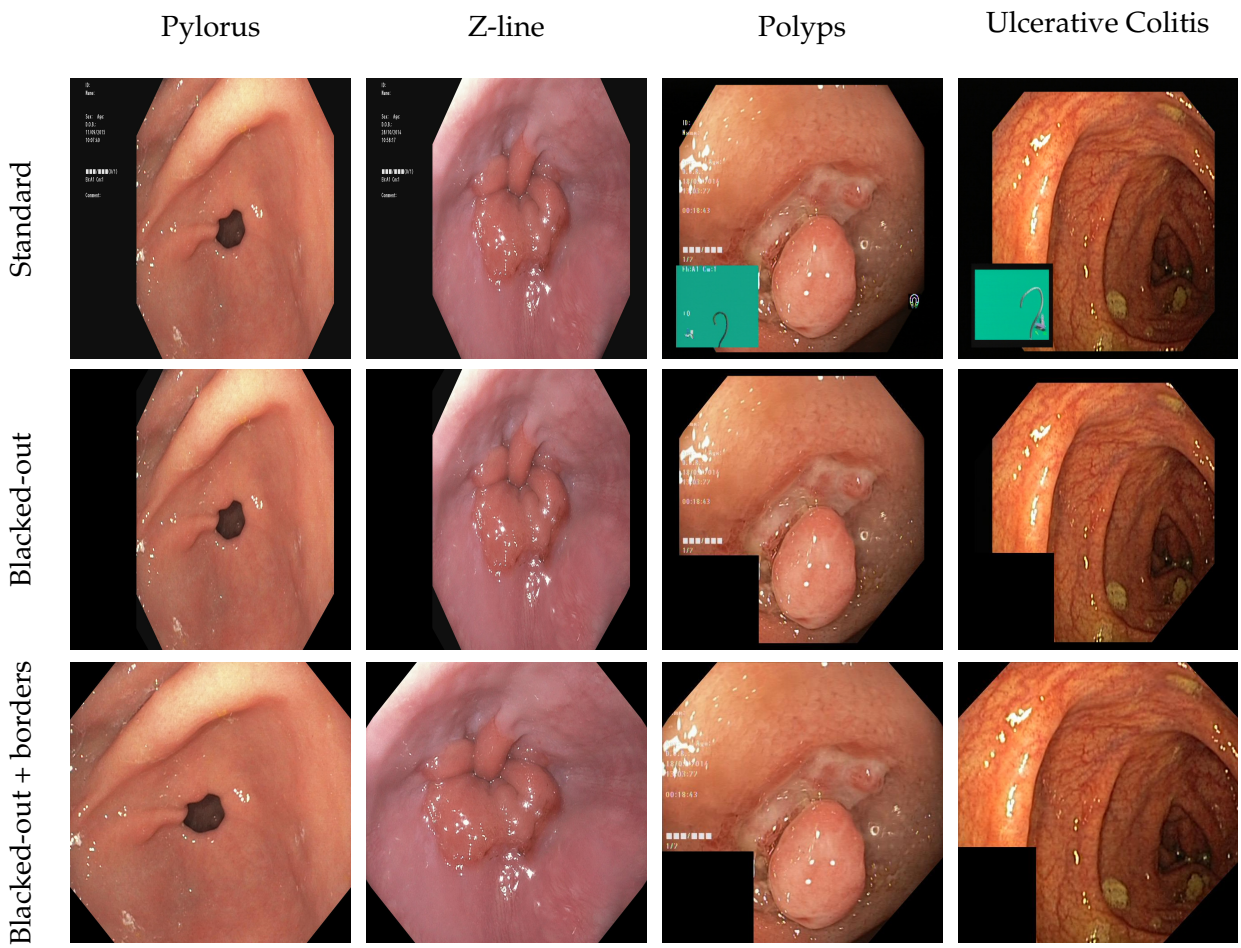


Figure 4.19: Four example images taken from the classes "pylorus", "z-line", "polyps" and "ulcerative coltiis", showing Kvasir (v2) after pre-processing.

4.4.6 Summary of Findings and Proposed Pre-processing Techniques

Based on the analysis done on each of the trained models in the previous section, we decided on two pre-processing techniques in which we hypothesized would lead to better classification results and generalizability. Firstly, looking at some of the confused activations, we saw that the green navigation box distracted from the actual features of some classes, resulting in either misclassifications or incorrect feature localizations. Prime examples of this can be seen in Figures 4.14d and 4.15e. Additionally, other artifacts located in the borders of the image (mostly text) played some role in some classification cases. Based on these findings, we decided just to black out these features, e.i., replace all appearances of the green navigation box and artifacts located in the outskirts of the image with a black cover. Note that we did not black out artifacts which overlaid parts of the image,

such as text and the small anchor located in the lower right corner of some images. This was the first of two pre-processing steps derived from the previously described analysis, which we decided to call the "blacked-out" pre-processing step.

Secondly, we noticed that borders themselves played some role in the misclassifications, specifically the transition from the colored image to the black border. This can be seen in Figures 4.14b and 4.17c, where we see that the highest activations are on the border itself, and not the class defining object. This border is present in all classes, albeit some to a lesser degree. This error in our networks leads to the second pre-processing step, which builds on the first. In addition to blacking out various artifacts, we also completely remove the vertical and horizontal borders of every image. We do not remove the diagonal borders due to the loss of image data that would occur. If we were to remove the diagonal borders, we would have to crop the rest of the image accordingly, resulting in potential loss of the class defining object, this is perhaps well suited for future work. This was the second and last pre-processing step derived from the analyses performed in the previous section, and we decided to call this pre-processing step "blacked out and borders removed".

We applied pre-processing techniques to the Kvasir (v2) dataset, leaving us with a total of three datasets used for comparison [44]. Examples from each class and pre-processing step can be seen in Figure 4.18 and Figure 4.19. With the two additional datasets, it was now time to train and re-evaluate each architecture using each of the pre-processed datasets. The result would be 15 trained models used for comparison. Each of the additional models was trained in the same way as we described in Section 4.1, using the hyperparameters shown in Table 4.3. This was done to make sure that the only variable in evaluating the models is due to the difference in the dataset used for training.

4.5 Results and Comparing New Visualizations Against Initial Results

Having trained and evaluated each of the five networks using the two pre-processed datasets, and re-analyzed each of them using the strategy described in Section 4.4. We present the results of what effect the pre-processing techniques had on the quality of each model. Firstly, by comparing the F1-score for each model trained on the pre-processed datasets against the models trained on the non-processed dataset (Table 4.6), we see that every model, except for Xception, improved between 1% to 2%. Furthermore, when introducing the additional 500 images from the CVC-982 dataset [16], the pre-processed models retain more of its score than its non-processed counterparts (Table 4.7). These results go to show that removing the different artifacts and borders part of Kvasir (v2) does give better training results in the form of metrics and

Kvasir (v2)	PREC	REC	SPEC	ACC	MCC	F1
VGG-16						
non processed	0.977	0.841	0.813	0.959	0.834	0.833
blacked out	0.979	0.856	0.833	0.963	0.854	0.852
blacked out + borders	0.977	0.842	0.816	0.960	0.839	0.838
VGG-19						
non processed	0.975	0.830	0.799	0.955	0.822	0.822
blacked out	0.976	0.837	0.811	0.958	0.833	0.833
blacked out + borders	0.976	0.838	0.813	0.959	0.835	0.836
ResNet-50						
non processed	0.980	0.864	0.841	0.965	0.860	0.860
blacked out	0.980	0.865	0.843	0.966	0.862	0.862
blacked out + borders	0.981	0.871	0.848	0.966	0.865	0.865
Inception (v3)						
non processed	0.975	0.830	0.802	0.956	0.825	0.825
blacked out	0.977	0.843	0.819	0.960	0.841	0.841
blacked out + borders	0.977	0.843	0.820	0.960	0.842	0.842
Xception						
non processed	0.980	0.859	0.838	0.964	0.858	0.858
blacked out	0.979	0.854	0.832	0.963	0.853	0.853
blacked out + borders	0.977	0.842	0.818	0.960	0.821	0.841

Table 4.6: A comparison of the evaluation results of all models trained on all versions of Kvasir (v2).

generalizability to new datasets. Now, there is the anomaly of the Xception architecture trained on the pre-processed datasets result in overall worse performance. The exact reason for this is hard to pinpoint from the metrics alone, but looking at some visualizations may shed some light on this topic. In the following few sections, we take a closer look at one of the previously analyzed image sets from each of the confused class pairs.

4.5.1 Comparing Dyed Resection Margin to Dyed Lifted Polyp

Figure 4.20 shows the extended analysis on the "dyed resection margin" image misclassified by the VGG-19 model in Section 4.4.1. Firstly, comparing how the prediction scores changed over each pre-processing step, we see that the network becomes increasingly confident in the correct classification. Looking at the visualizations, we see that when targeting the correct class ("dyed resection margin"), the activations continue to focus on the resection mark. Targeting the wrong class of "dyed lifted polyp", we see that the activations move as artifacts are removed from the image,

Kvasir (v2) + CVC-968	PREC	REC	SPEC	ACC	MCC	F1
VGG-16						
non processed	0.966	0.770	0.743	0.940	0.791	0.768
blacked out	0.969	0.786	0.748	0.944	0.808	0.786
blacked out + borders	0.973	0.812	0.789	0.953	0.823	0.812
VGG-19						
non processed	0.966	0.771	0.739	0.940	0.784	0.766
blacked out	0.967	0.778	0.748	0.941	0.794	0.773
blacked out + borders	0.970	0.805	0.777	0.948	0.813	0.801
ResNet-50						
non processed	0.975	0.826	0.805	0.956	0.837	0.827
blacked out	0.978	0.843	0.823	0.961	0.850	0.843
blacked out + borders	0.978	0.850	0.824	0.961	0.848	0.843
Inception (v3)						
non processed	0.971	0.803	0.774	0.949	0.806	0.801
blacked out	0.974	0.819	0.795	0.954	0.825	0.819
blacked out + borders	0.974	0.821	0.797	0.955	0.826	0.820
Xception						
non processed	0.976	0.836	0.816	0.959	0.843	0.838
blacked out	0.976	0.833	0.811	0.958	0.838	0.834
blacked out + borders	0.973	0.813	0.789	0.953	0.821	0.814

Table 4.7: A comparison of the evaluation results of all models trained on all versions of Kvasir (v2).

ending with heavy activations on the overlaid text. This provides further proof that the removal of the text would most likely further increase the model's confidence. Nevertheless, we managed to go from an incorrect classification to a near 100% certainty with the application of two simple pre-processing steps.

4.5.2 Comparing Esophagitis to Z-line

Figure 4.21 shows the extended analysis on the "esophagitis" image misclassified by the ResNet-50 model in Section 4.4.2. Looking at the visualizations for the model trained on the "blacked out" pre-processed dataset (Figures 4.21c and 4.21d), we see an improvement in the correct class score. However, when applying the more extensive pre-processing step of "blacked out and borders removed", the class score drops to a mere 5%, reverting the previous improvement. Looking at the visualizations in Figure 4.21f, we see that the high score of "z-line" is mostly due to the

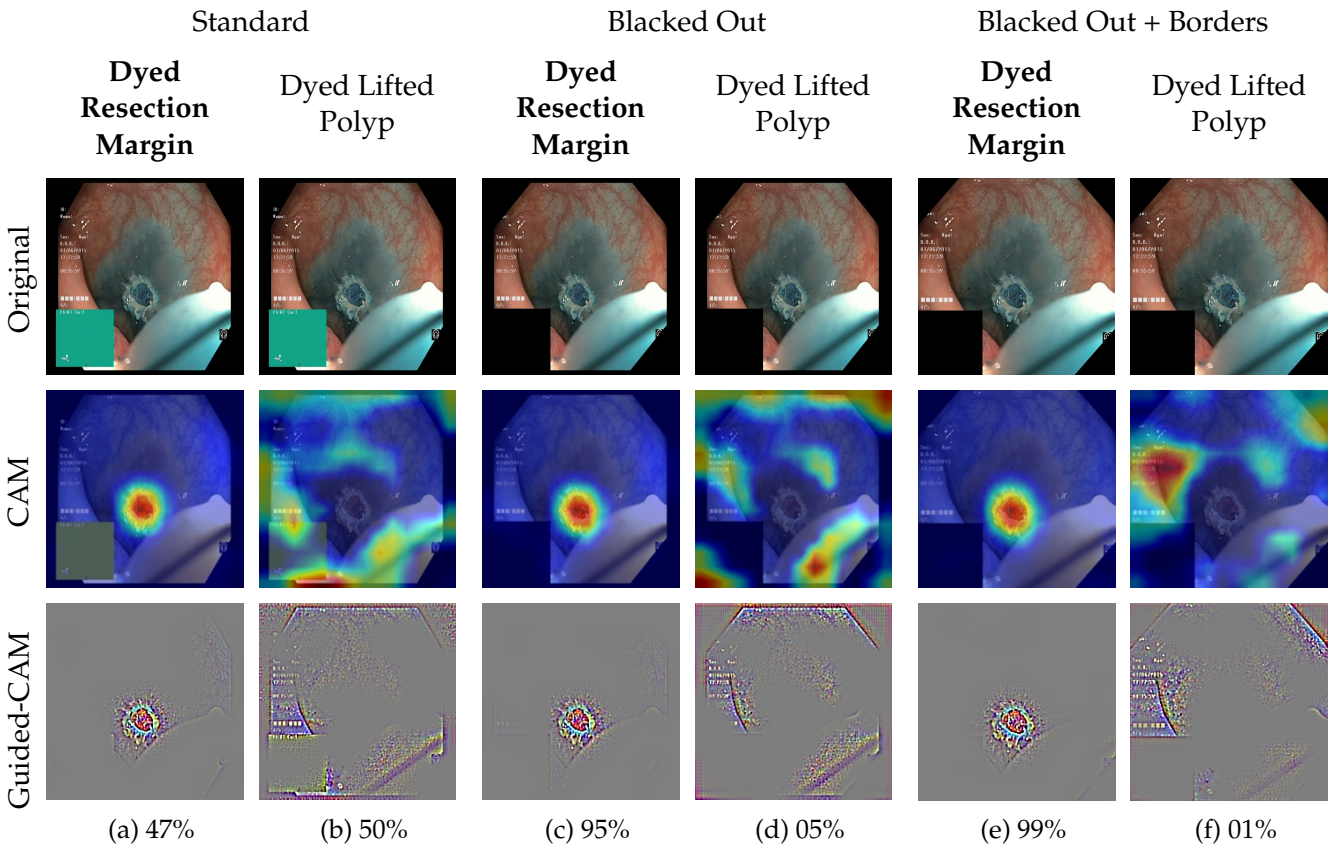


Figure 4.20: A collection of images showing the how activations changed between the various models trained on the standard, blacked out and blacked out and borders removed Kvasir (v2) datasets. Each column corresponds to visualizing for the label located over each column, with the predicted probability listed below. The visualizations in this figure correspond to the case where a VGG-19 based network misclassified an image as "dyed lifted polyp" when the actual class was "dyed resection margin".

presence of black corner borders. This provides further evidence that the removal of these corners would most likely rectify this misclassification.

4.5.3 Comparing Cecum to Ulcerative Colitis

Figure 4.22 shows the extended analysis on the "cecum" image misclassified by the ResNet-50 model in Section 4.4.3. Looking at the original activations for the correct class (Figures 4.22a), we see that the network mostly activates on the navigation box and overlaid text. After applying the "blacked out" pre-processing step (Figure 4.22c), we see that this attention then moved the mostly to activate on the left black border, which drops the confidence of the correct class down to 1%. Applying the final pre-processing step, "blacked out and borders removed", rectifies this

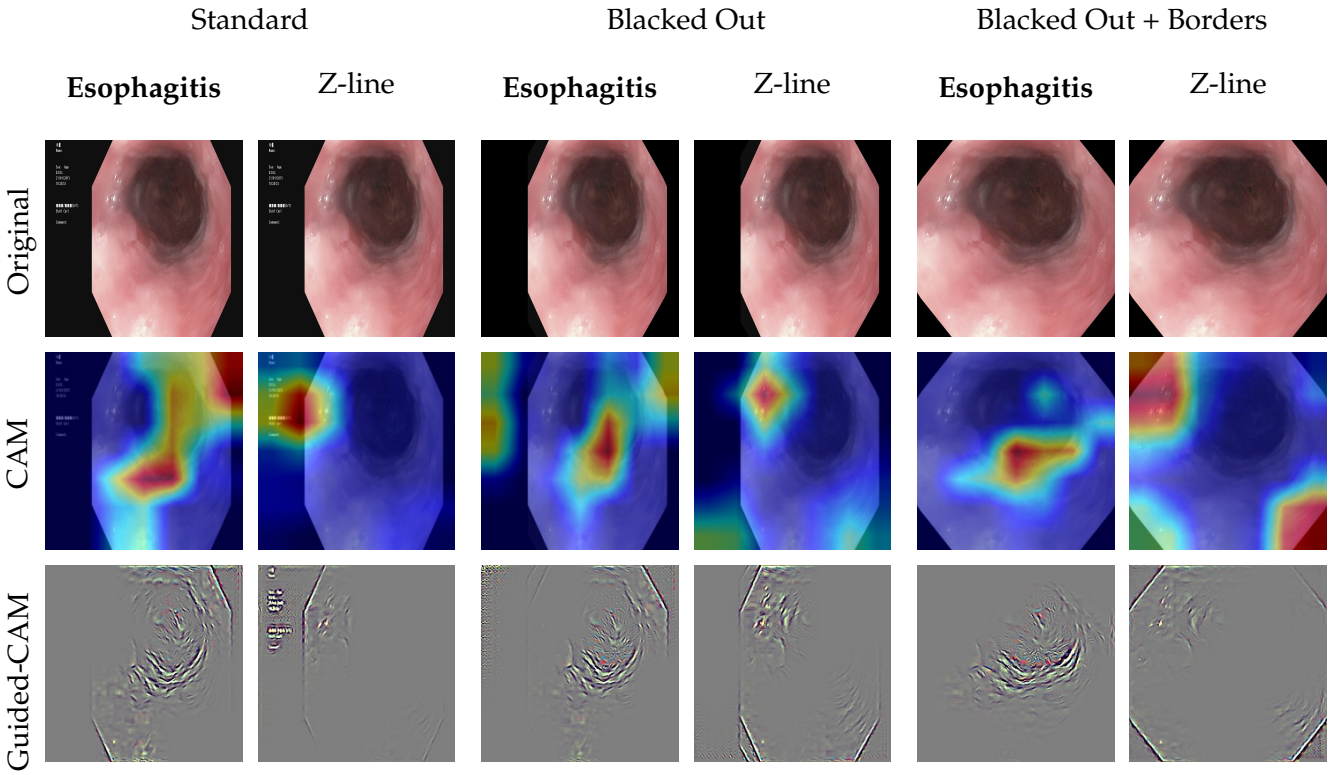


Figure 4.21: A collection of images showing the how activations changed between the various models trained on the standard, blacked out and blacked out and borders removed Kvasir (v2) datasets. Each column corresponds to visualizing for the label located over each column, with the predicted probability listed below. The visualizations in this figure correspond to the case where a ResNet-50 based network misclassified an image as "z-line" when the actual class was "esophagitis".

misclassification (Figure 4.22e). However, its activations focus mostly on the upper right black corner. In a slightly different case from the last, the "blacked out" pre-processing made the network perform worse, while the more extensive pre-processing step led to better classification, yet focuses on the wrong image features.

4.5.4 Comparing Polyp to Cecum

Figure 4.23 shows the extended analysis on the "polyp" image misclassified by the Xception model in Section 4.4.4. This case differentiates itself from the rest, as both pre-processing steps led to considerably worse classification results. Looking at the visualizations for the non-processed Kvasir (v2) (Figures 4.23a and 4.23b), we see that the network does, in fact, detect the polyp. Yet, the network still manages to misclassify it as "cecum". After applying the "blacked out" pre-processing step, we

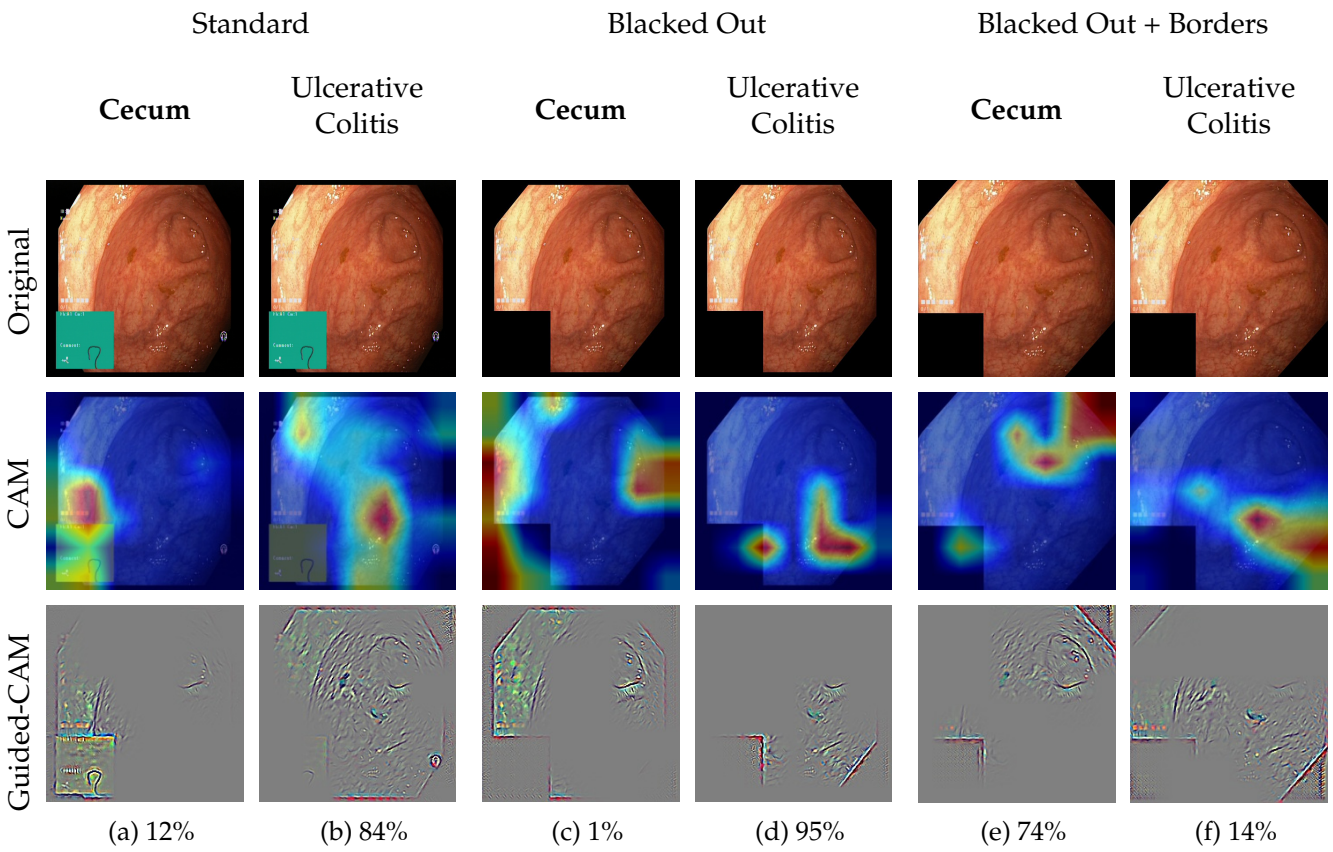


Figure 4.22: A collection of images showing the how activations changed between the various models trained on the standard, blacked out and blacked out and borders removed Kvasir (v2) datasets. Each column corresponds to visualizing for the label located over each column, with the predicted probability listed below. The visualizations in this figure correspond to the case where a ResNet-50 based network misclassified an image as "ulcerative colitis" when the actual class was "cecum".

see that the network still detects the polyp, yet the score drops to a 4% confidence. Applying the last pre-processing step of "blacked out and borders removed", we see that the classification then dropped to 0%, with the network 100% confident that this image is part of the "cecum" class. In conclusion, this was a strange case, but it seems like the artifacts play into most of the problems here. It would be interesting to see how the removal of text and complete removal of black borders would change this classification.

4.5.5 Comparing Ulcerative Colitis to Polyp

Figure 4.24 shows the extended analysis on the "ulcerative colitis" image misclassified by the ResNet-50 model in Section 4.4.5. Looking at the visualizations from the initial analysis (Figures 4.24a and 4.24b), we see

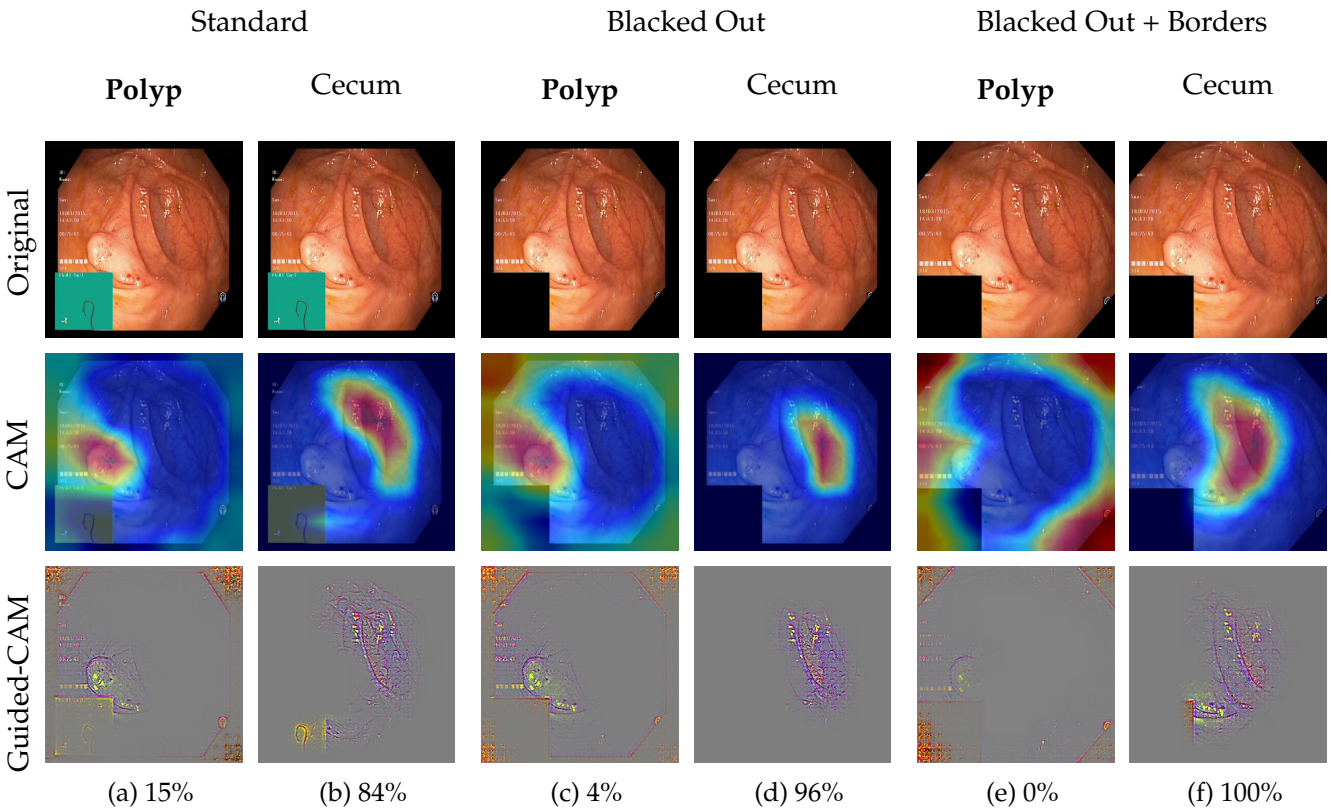


Figure 4.23: A collection of images showing the how activations changed between the various models trained on the standard, blacked out and blacked out and borders removed Kvasir (v2) datasets. Each column corresponds to visualizing for the label located over each column, with the predicted probability listed below. The visualizations in this figure correspond to the case where a Xception based network misclassified an image as "cecum" when the actual class was "polyp".

the "ulcerative colitis" score was mostly due to the borders and navigation box, and the "polyp" score was based on the change in depth of the image. Firstly, after applying the first pre-processing step of "blacked out", we see that the activations for "ulcerative colitis" shift to the reflections in the right region of the image, which decreases the score to a 9% confidence in the correct class. Applying the "blacked out and borders removed" processing step rectified the misclassification, but similar to that of the "cecum" case, the network bases this prediction on the lower right corner. In conclusion, we see that the black corners should be removed for both classification and localization reasons, as even though the network might correctly classify an image, it might not be due to the correct features.

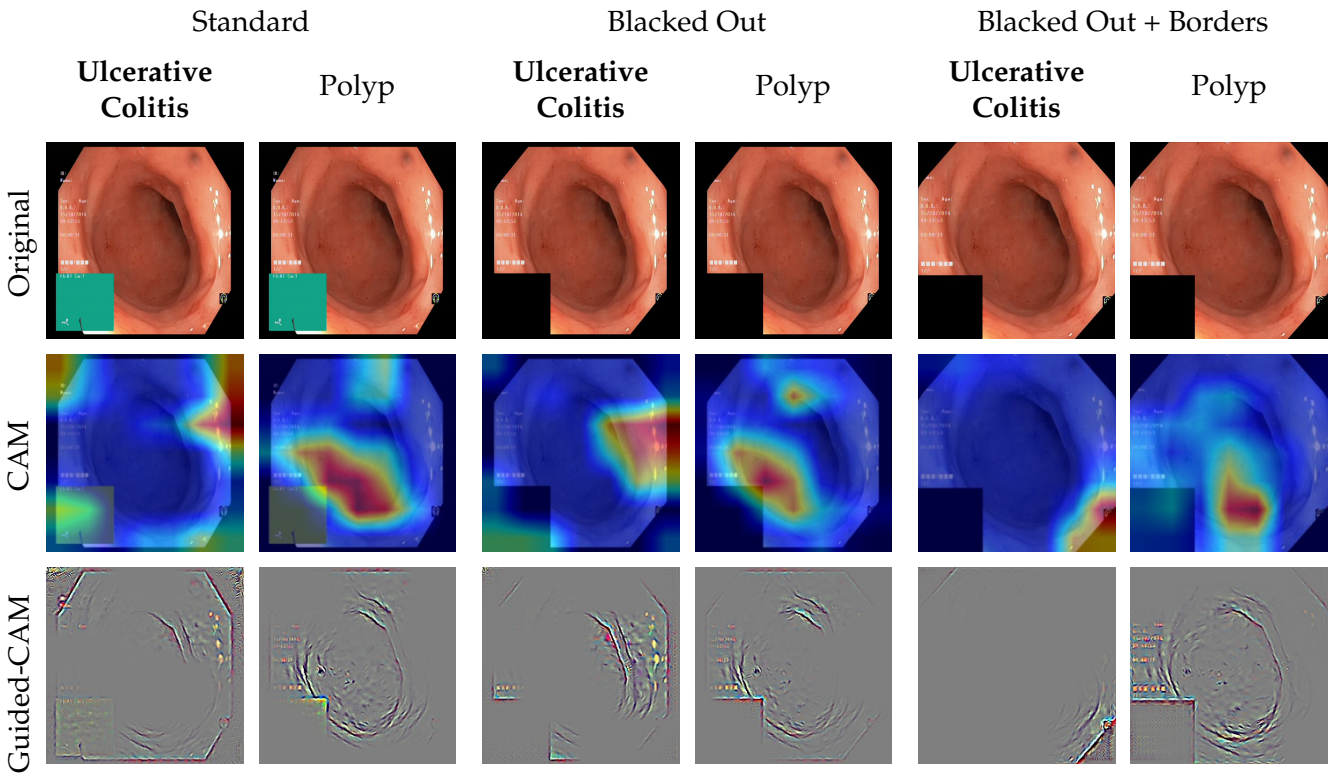


Figure 4.24: A collection of images showing the how activations changed between the various models trained on the standard, blacked out and blacked out and borders removed Kvasir (v2) datasets. Each column corresponds to visualizing for the label located over each column, with the predicted probability listed below. The visualizations in this figure correspond to the case where a ResNet-50 based network misclassified an image as "polyp" when the actual class was "ulcerative colitis".

4.6 Summary

Pre-processing of data before training deep neural networks is a common practice among high achieving models, but finding what pre-processing steps to apply to every piece of data is not always clear. Through the use of our CNN visualization techniques, we gain some insight into what areas of a given image is highly activated for a variety of different deep learning architectures. In this chapter, we look at using three visualization techniques; grad-CAM, guided backpropagation, and guided grad-CAM, to get a better understanding of models trained on the Kvasir (v2) dataset with the purpose of improving the quality and generalizability. To do this, we use our Mimir system [44–46] (described in the previous chapter) to analyze five different architectures; VGG-16, VGG-19, Inception (v3), Xception and ResNet-50, trained on Kvasir (v2) to derive two pre-processing steps, which are then applied to each of the five models and compared against their initial results. This work relates to the

third and final research objective, where we wanted to see how a deeper understanding of the analysis done by a CNN could lead to better training results. The experiments conducted in this chapter relate back to our last research objective, where we wanted to test our system to see how getting a better understanding of the decision process to a deep neural network could result in knowledge to better improve the performance of existing models.

After conducting our experiments, we found that all models except for one improved after applying the derived pre-processing steps to training. This proves that having a better understanding of how trained models "see" a given class may be used to improve the quality of that model.

Chapter 5

Conclusion and Further Work

5.1 Summary

Nowadays, deep neural networks are used in a wide variety of different fields ranging from the automatic detection of fraud to automatically translating text from images. However, these methods are rarely understood and are often treated as if they were a "black box". This common fault may not be much of an issue when dealing with problems of little consequence, such as the automatic distinction between cats and dogs. But, when applying these methods to problems where a mistake may result in life-altering consequences, we often need some rationale behind why these algorithms suggest a given result. As a consequence, less complex analytical models are often preferred, even though they might be less accurate. With these open problems, we researched and developed a system for automatic endoscopy reporting called Mimir, which focuses on transparency and understanding of the analysis done by a deep CNN used for the detection and diagnosis of disease found in the GI tract. This improved understanding of the Mimir's analysis is then used to generate standard compliant endoscopy reports, which the user may edit and format as they please.

To test the usefulness of this system, we performed a case study using Mimir to see how a better understanding of a deep neural networks internal process may lead to finding methods of improving a given models classification result. This was done through the analysis of five models, each based on a different CNN architecture, trained on the endoscopy image dataset called Kvasir [76]. By analyzing the last convolutional layer of each network, we derived two pre-processing techniques which we hypothesized would increase model performance. Our results showed an overall increase of about 2% (looking at the calculated F1-score), except for one network, which saw a decrease in performance. Additionally, we saw a general increase in localization performance based on a selection of images. However, we were not able to conclusively validate the localization performance of each network, seeing as we did not have access to an annotated masking dataset for Kvasir.

5.2 Contributions

As discussed in the problem statement of this thesis, we derived three research objectives. Below, we restate each objective together with a description of how our work solves the stated problems.

Objective 1 *Research and develop a system which gives non-technical users a better understanding of why a neural network presents a given result. This system should be aimed at medical doctors conducting examinations and documentation abnormalities found in the GI tract.*

This objective is supported by the development of Mimir [45, 46], which provides a tool for dissecting the internal layers of a deep CNN. Using this tool, a doctor may verify that the diagnosis suggested by the system is in fact due to the detection of said disease, and not due to artifacts or noise commonly found in medical images. Also, the system allows for direct targeting of different classes, showing what regions of a given image directly contributes to the assigned score.

Objective 2 *Provide a proof-of-concept implementation of automatic GI report generation based on the findings of automatic analysis done through the use of a deep neural network.*

This objective is supported by the report generation tool included in Mimir [45, 46], which suggests relevant images based on a diagnosis proposed by the system. As stated in the objective, this is currently a proof-of-concept, meaning it is expected to be improved through future work.

Objective 3 *Use various visualization techniques to get a better understanding of the internal working of a deep neural network. This newly gained knowledge should be used in the development of pre-processing steps with the purpose of training quality and robust analytical models based on deep learning.*

This last objective is supported by our use of Mimir to analyze five neural network based models, each using a different CNN architecture, with the purpose of finding faults in its training. This was done using Mimir's neural network dissection tool, where we inspected what regions of a given image correlated to the incorrect classification score. Based on the performed analysis, we derived two pre-processing steps applied to Kvasir (v2) [76] dataset, which showed to improve the classification score of all models except one. This objective is also supported by a published paper [44], where we showcase part of the experiments conducted over the course of this thesis.

Through the work produced in this thesis, we learned that using neural network based visualizations may provide sufficient knowledge into what pre-processing steps may lead to improved classification performance. Specifically, we improved the performance of a deep neural network trained to detect disease and anatomy of the GI tract.

Each objective is supported by published papers, which have also been included in the Appendix of this thesis. Paper B.1 [46] and Paper B.2 [45] relate to the first two objectives, and Paper B.3 [44] relates to the final objective.

5.3 Future Work

Getting a better understanding of the internal workings of a deep neural network is an essential first step in increasing trust and acceptance in systems based on this technology. In Chapter 3, we presented a system for analysis of such deep neural networks with the purpose of using this information in the reporting of GI disease. For future work, we would like to expand on the report generation portion of the system by adding support for features such as automatic text suggestions, support for different reporting templates, and a more robust system for producing and exporting generated reports. Even though doctors listed an automatic text generation as a low priority, it would still improve on the implementation of the MST as suggested by the ESGE.

In addition to improving the reporting system of Mimir, we see that additional experiments using the neural network dissection tool could be useful to the quality of Kvasir. Through further inspection, we see other pre-processing steps that may give train higher quality models. In addition to the proposed pre-processing steps, we could remove the text located on some of the images, entirely remove the anchor found in the lower right corner of some images. Lastly, thoroughly remove all borders by cropping the entire image, leaving no black borders of any kind (although this would require more than a simple pre-processing step as we would not like to crop out the class defining object).

Bibliography

[1] Lars Aabakken, Alan N Barkun, Peter B Cotton, Evgeny Fedorov, Masayuki A Fujino, Ekaterina Ivanova, Shin-ei Kudo, Konstantin Kuznetzov, Thomas Lange, Koji Matsuda et al. 'Standardized endoscopic reporting'. In: ***Journal of gastroenterology and hepatology*** 29.2 (2014), pp. 234–240.

[2] Airbnb. ***Enzyme***. 2018. URL: https://github.com/airbnb/enzyme.

[3] Dinu A.J, R Ganesan and F. Knight Joseph. 'A study on Deep Machine Learning Algorithms for diagnosis of diseases'. In: 2017.

[4] Davide Bacciu, Paulo J. G. Lisboa, José D. Martín, Ruxandra Stoean and Alfredo Vellido. 'Bioinformatics and Medicine in the Era of Deep Learning'. In: ***CoRR*** abs/1802.09791 (2018). arXiv: 1802.09791. URL: http://arxiv.org/abs/1802.09791.

[5] Nancy N Baxter, Rinku Sutradhar, Shawn S Forbes, Lawrence F Paszat, Refik Saskin and Linda Rabeneck. 'Analysis of Administrative Data Finds Endoscopist Quality Measures Associated With Postcolonoscopy Colorectal Cancer'. In: ***Gastroenterology*** 140.1 (Apr. 2018), pp. 65–72. ISSN: 0016-5085. DOI: 10.1053/j.gastro.2010.09.006. URL: http://dx.doi.org/10.1053/j.gastro.2010.09.006.

[6] Daphnée Beaulieu, Alan Barkun and Myriam Martel. 'Quality audit of colonoscopy reports amongst patients screened or surveilled for colorectal neoplasia'. In: ***World Journal of Gastroenterology : WJG*** 18.27 (July 2012), pp. 3551–3557. ISSN: 1007-9327. DOI: 10.3748/wjg.v18.i27.3551. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3400856/.

[7] R. Bhardwaj, A. R. Nambiar and D. Dutta. 'A Study of Machine Learning in Healthcare'. In: ***2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)***. Vol. 2. July 2017, pp. 236–241. DOI: 10.1109/COMPSAC.2017.164.

[8] Michael Bretthauer, Lars Aabakken, Evelien Dekker, Michal F Kaminski, Thomas Rösch, Rolf Hultcrantz, Stepan Suchanek, Rodrigo Jover, Ernst J Kuipers, Raf Bisschops et al. 'Requirements and standards facilitating quality improvement for reporting systems in gastrointestinal endoscopy: European Society of Gastrointestinal Endoscopy (ESGE) Position Statement'. In: ***Endoscopy*** 48.3 (2016), pp. 291–4.

[9] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm and Noemie Elhadad. 'Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission'. In: ***Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining***. KDD '15. Sydney, NSW, Australia: ACM, 2015, pp. 1721–1730. ISBN: 978-1-4503-3664-2. DOI: 10.1145/2783258.2788613. URL: http://doi.acm.org/10.1145/2783258.2788613.

[10] Hongming Chen, Ola Engkvist, Yinhai Wang, Marcus Olivecrona and Thomas Blaschke. 'The rise of deep learning in drug discovery'. In: ***Drug Discovery Today*** (2018). ISSN: 1359-6446. DOI: https://doi.org/10.1016/j.drudis.2018.01.039. URL: http://www.sciencedirect.com/science/article/pii/S1359644617303598.

[11] Shawn Chen and Douglas K. Rex. 'Endoscopist Can Be More Powerful than Age and Male Gender in Predicting Adenoma Detection at Colonoscopy'. In: ***The American Journal of Gastroenterology*** 102 (2007), pp. 856–861.

[12] François Chollet et al. 'Keras: Deep learning library for theano and tensorflow'. In: ***URL: https://keras. io/k*** (2015).

[13] François Chollet. 'Xception: Deep Learning with Depthwise Separable Convolutions'. In: ***CoRR*** abs/1610.02357 (2016). arXiv: 1610.02357. URL: http://arxiv.org/abs/1610.02357.

[14] Susan G Coe, Chakri Panjala, Michael G Heckman, Mihir Patel, Bashar J Qumseya, Yize R Wang, Benjamin Dalton, Philip Tran, William Palmer, Nancy Diehl, Michael B Wallace and Massimo Raimondo. 'Quality in colonoscopy reporting: An assessment of compliance and performance improvement'. In: ***Digestive and Liver Disease*** 44.8 (Apr. 2018), pp. 660–664. ISSN: 1590-8658. DOI: 10.1016/j.dld.2012.03.022. URL: http://dx.doi.org/10.1016/j.dld.2012.03.022.

[15] R. Collobert, K. Kavukcuoglu and C. Farabet. 'Torch7: A Matlab-like Environment for Machine Learning'. In: ***BigLearn, NIPS Workshop***. 2011.

[16] 'Comparative Validation of Polyp Detection Methods in Video Colonoscopy: Results from the MICCAI 2015 Endoscopic Vision Challenge'. In: ***IEEE Transactions on Medical Imaging*** (99 2017).

[17] Gregory F. Cooper, Constantin F. Aliferis, Richard Ambrosino, John Aronis, Bruce G. Buchanan, Richard Caruana, Michael J. Fine, Clark Glymour, Geoffrey Gordon, Barbara H. Hanusa, Janine E. Janosky, Christopher Meek, Tom Mitchell, Thomas Richardson and Peter Spirtes. 'An evaluation of machine-learning methods for predicting pneumonia mortality'. In: ***Artificial Intelligence in Medicine*** 9.2 (1997), pp. 107–138. ISSN: 0933-3657. DOI: https://doi.org/10.1016/S0933-3657(96)00367-3. URL: http://www.sciencedirect.com/science/article/pii/S0933365796003673.

[18] ***CrowdMM '14: Proceedings of the 2014 International ACM Workshop on Crowdsourcing for Multimedia***. Orlando, Florida, USA: ACM, 2014. ISBN: 978-1-4503-3128-9.

[19] G. Cybenko. 'Approximation by superpositions of a sigmoidal function'. In: ***Mathematics of Control, Signals and Systems*** 2.4 (Dec. 1989), pp. 303–314. ISSN: 1435-568X. DOI: 10.1007/BF02551274. URL: https://doi.org/10.1007/BF02551274.

[20] M Delvaux, L.Y Korman, J.R Armengol-Miro, M Crespi, O Cass, F Hagenmüller and F.M Zwiebel. 'The minimal standard terminology for digestive endoscopy: introduction to structured reporting'. In: ***International Journal of Medical Informatics*** 48.1 (1998), pp. 217–225. ISSN: 1386-5056.

[21] Peter Denning, Douglas E. Comer, David Gries, Michael C. Mulder, Allen B. Tucker, A. Joe Turner and Paul R. Young. 'Computing as a discipline: preliminary report of the ACM task force on the core of computer science'. In: ***Proceedings of the Nineteenth SIGCSE Technical Symposium on Computer Science Education*** 20.1 (1988), p. 41. ISSN: 00978418. DOI: 10.1145/52964.52975. URL: http://doi.acm.org/10.1145/52964.52975%0Ahttp://doi.acm.org/10.1145/52965.52975.

[22] Timothy Dozat. 'Incorporating Nesterov Momentum into Adam'. In: 2015.

[23] John Duchi, Elad Hazan and Yoram Singer. 'Adaptive Subgradient Methods for Online Learning and Stochastic Optimization'. In: ***J. Mach. Learn. Res.*** 12 (July 2011), pp. 2121–2159. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1953048.2021068.

[24] Nicola Eastaff-Leung, Nicholas Mabarrack, Angela Barbour, Adrian Cummins and Simon Barry. 'Foxp3+ Regulatory T Cells, Th17 Effector Cells, and Cytokine Environment in Inflammatory Bowel Disease'. In: ***Journal of Clinical Immunology*** 30.1 (Jan. 2010), pp. 80–89. ISSN: 1573-2592. DOI: 10.1007/s10875-009-9345-1. URL: https://doi.org/10.1007/s10875-009-9345-1.

[25] Dumitru Erhan, Yoshua Bengio, Aaron Courville and Pascal Vincent. 'Visualizing higher-layer features of a deep network'. In: ***Bernoulli*** 1341 (2009), pp. 1–13. URL: http://igva2012.wikispaces.asu.edu/file/view/Erhan+2009+Visualizing+higher+layer+features+of+a+deep+network.pdf.

[26] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau and Sebastian Thrun. 'Dermatologist-level classification of skin cancer with deep neural networks'. In: ***Nature*** 542.7639 (Feb. 2017), pp. 115–118. ISSN: 0028-0836. DOI: 10.1038/nature21056. URL: http://dx.doi.org/10.1038/nature21056%2010.1038/nature21056%20http://www.nature.com/doifinder/10.1038/nature21056.

[27] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau and Sebastian Thrun. 'Dermatologist-level classification of skin cancer with deep neural networks'. In: ***Nature*** 542.7639 (2017), pp. 115–118. ISSN: 0028-0836. DOI: 10.1038/nature21056. URL: http://dx.doi.org/10.1038/nature21056%2010.1038/nature21056%20http://www.nature.com/doifinder/10.1038/nature21056.

[28] ***Estimated Cancer Incidence, Mortality and Prevalence Worldwide in 2012***. [last visited, May. 12, 2018]. 2012. URL: http://globocan.iarc.fr/Pages/fact_sheets_population.aspx.

[29] Facebook. ***Flux***. 2018. URL: https://facebook.github.io/flux/.

[30] Facebook. ***Jest***. 2018. URL: https://facebook.github.io/jest/.

[31] Facebook. ***React***. 2018. URL: https://reactjs.org/.

[32] Li Fei-Fei, R. Fergus and P. Perona. 'One-shot learning of object categories'. In: ***IEEE Transactions on Pattern Analysis and Machine Intelligence*** 28.4 (Apr. 2006), pp. 594–611. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2006.79.

[33] Deborah A Fisher, John T Maple, Tamir Ben-Menachem, Brooks D Cash, G Anton Decker, Dayna S Early, John A Evans, Robert D Fanelli, Norio Fukami, Joo Ha Hwang, Rajeev Jain, Terry L Jue, Khalid M Khan, Phyllis M Malpas, Ravi N Sharaf, Amandeep K Shergill and Jason A Dominitz. 'Complications of colonoscopy'. In: ***Gastrointestinal Endoscopy*** 74.4 (Oct. 2011), pp. 745–752. ISSN: 0016-5107. DOI: 10.1016/j.gie.2011.07.025. URL: http://dx.doi.org/10.1016/j.gie.2011.07.025.

[34] Flask. ***Flask***. 2018. URL: http://flask.pocoo.org/.

[35] Christopher J Fyock and Peter V Draganov. 'Colonoscopic polypectomy and associated techniques'. In: ***World Journal of Gastroenterology : WJG*** 16.29 (Aug. 2010), pp. 3630–3637. ISSN: 1007-9327. DOI: 10.3748/wjg.v16.i29.3630. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2915422/.

[36] Lauren B Gerson, Jeffrey Tokar, Michael Chiorean, Simon Lo, G Anton Decker, David Cave, Doumit BouHaidar, Daniel Mishkin, Charles Dye, Oleh Haluszka, Jonathan A Leighton, Alvin Zfass and Carol Semrad. 'Complications Associated With Double Balloon Enteroscopy at Nine US Centers'. In: ***Clinical Gastroenterology and Hepatology*** 7.11 (Nov. 2009), 1177–1182.e3. ISSN: 1542-3565. DOI: 10.1016/j.cgh.2009.07.005. URL: http://dx.doi.org/10.1016/j.cgh.2009.07.005.

[37] Gregory G. Ginsberg. 'Risks of Colonoscopy and Polypectomy'. In: ***Techniques in Gastrointestinal Endoscopy*** 10.1 (2008). Volume 2. Risks of Endoscopy and the Endoscopist, the Endoscopy Staff, and the Patient, pp. 7–13. ISSN: 1096-2883. DOI: https://doi.org/10.1016/j.tgie.2007.08.005. URL: http://www.sciencedirect.com/science/article/pii/S1096288307000484.

[38] Google. ***Angular***. 2018. URL: https://angular.io.

[39] H. Greenspan, B. van Ginneken and R. 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.

[40] Marcel Groenen, Ernst Kuipers, Gerard van Berge Henegouwen, Paul Fockens and Rob Ouwendijk. 'Computerisation of endoscopy reports using standard reports and text blocks'. In: ***The Netherlands journal of medicine*** (2006).

[41] Nigel H. ***The Crohnoid Blog***. 2015. URL: http : / / www . wrestlingtheoctopus.com/the-a-to-z-of-my-crohns/.

[42] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. 'Deep Residual Learning for Image Recognition'. In: ***CoRR*** abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[43] Steven Hicks. ***Master Training Scripts***. https : / / github . com / Stevenah/master-training-scripts. 2018.

[44] Steven Hicks, Michael Riegler, Pogorelov Konstantin, Kim V. Ånonsen, Thomas de Lange, Dag Johansen, Mattis Jeppsson, Kristin Ranheim Randel, Sigrun Eskeland and Pål Halvorsen. ***Dissecting Deep Neural Networksfor Better Medical Image Classification andClassification Understanding***. 2018.

[45] Steven Hicks, Michael Riegler, Pogorelov Konstantin, Thomas de Lange, Dag Johansen, Mattis Jeppsson, Kristin Ranheim Randel, Sigrun Eskeland and Pål Halvorsen. 'Comprehensible Reasoning and Automated Reporting ofMedical Examinations Based on Deep Learning Analysis'. In: ***InProceedings of 9th ACM Multimedia Systems Conference, Amsterdam, Netherlands, June 12–15, 2018 (MMSys'18)***. ACM, 2018. DOI: 10.1145/3204949.3208113. URL: https://doi.org/10.1145/3204949.3208113.

[46] Steven Hicks, Michael Riegler, Pogorelov Konstantin, Thomas de Lange, Dag Johansen, Mattis Jeppsson, Kristin Ranheim Randel, Sigrun Eskeland and Pål Halvorsen. 'Mimir: An Automatic Reporting and Reasoning System for DeepLearning based Analysis in the Medical Domain'. In: ***InProceedings of 9th ACM Multimedia Systems Conference, Amsterdam, Netherlands, June 12–15, 2018 (MMSys'18)***. ACM, 2018. DOI: 10.1145/3204949.3208129. URL: https://doi.org/10.1145/3204949.3208129.

[47] Robert J Hilsden, Alaa Rostom, Catherine Dubé, Darlene Pontifex, S Elizabeth McGregor and Ronald J Bridges. 'Development and implementation of a comprehensive quality assurance program at a community endoscopy facility'. In: ***Canadian Journal of Gastroenterology*** 25.10 (Oct. 2011), pp. 547–554. ISSN: 0835-7900. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3206550/.

[48] Geoffrey E. Hinton, Simon Osindero and Yee-Whye Teh. 'A Fast Learning Algorithm for Deep Belief Nets'. In: ***Neural Comput.*** 18.7 (July 2006), pp. 1527–1554. ISSN: 0899-7667. DOI: 10.1162/neco.2006.18.7.1527. URL: http://dx.doi.org/10.1162/neco.2006.18.7.1527.

[49] Najihah Ibrahim, Nur Shazwani Md. Akhir and Fadratul Hafinaz Hassan. 'Predictive analysis effectiveness in determining the epidemic disease infected area'. In: ***AIP Conference Proceedings*** 1891.1 (2017), p. 020064. DOI: 10.1063/1.5005397. eprint: https://aip.scitation.org/doi/pdf/10.1063/1.5005397. URL: https://aip.scitation.org/doi/abs/10.1063/1.5005397.

[50] X. Jia and M. Q. H. Meng. 'A deep convolutional neural network for bleeding detection in Wireless Capsule Endoscopy images'. In: ***2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)***. Aug. 2016, pp. 639–642. DOI: 10.1109/EMBC.2016.7590783.

[51] Vincent de Jonge, Jerome Sint Nicolaas, Djuna L Cahen, Willem Moolenaar, Rob J.Th. Ouwendijk, Thjon J Tang, Antonie J P van Tilburg, Ernst J Kuipers and Monique E van Leerdam. 'Quality evaluation of colonoscopy reporting and colonoscopy performance in daily clinical practice'. In: ***Gastrointestinal Endoscopy*** 75.1 (Apr. 2018), pp. 98–106. ISSN: 0016-5107. DOI: 10.1016/j.gie.2011.06.032. URL: http://dx.doi.org/10.1016/j.gie.2011.06.032.

[52] Andrej Karpathy, Justin Johnson and Fei-Fei Li. 'Visualizing and Understanding Recurrent Networks'. In: ***CoRR*** abs/1506.02078 (2015). arXiv: 1506.02078. URL: http://arxiv.org/abs/1506.02078.

[53] Ji Hyun Kim, Jin Ki Hwang, Juhyung Kim, Sehe Dong Lee, Beom Jae Lee, Jae Seon Kim and Young-Tae Bak. 'Endoscopic findings around the gastroesophageal junction: an experience from a tertiary hospital in Korea'. In: ***The Korean Journal of Internal Medicine*** 23.3 (Sept. 2008), pp. 127–133. ISSN: 1226-3303. DOI: 10.3904/kjim.2008.23.3.127. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2686963/.

[54] Diederik P. Kingma and Jimmy Ba. 'Adam: A Method for Stochastic Optimization'. In: ***CoRR*** abs/1412.6980 (2014). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

[55] Louis Y Korman. 'Standardization in Endoscopic Reporting: Implications for Clinical Practice and Research'. In: ***Journal of Clinical Gastroenterology*** 28.3 (1999). ISSN: 0192-0790. URL: https://journals.lww.com/jcge/Fulltext/1999/04000/Standardization%7B%5C_%7Din%7B%5C_%7DEndoscopic%7B%5C_%7DReporting%7B%5C_%7D.6.aspx.

[56] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. 'ImageNet Classification with Deep Convolutional Neural Networks'. In: ***Advances in Neural Information Processing Systems 25***. Ed. by F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/

paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[57] K Kuhn, W Gaus, JG Wechsler, P Janowitz, J Tudyka, W Kratzer, W Swobodnik and H Ditschuneit. 'Structured reporting of medical findings: evaluation of a system in gastroenterology'. In: ***Methods of information in medicine*** 31.04 (1992), pp. 268–274.

[58] Yann LeCun, Patrick Haffner, Léon Bottou and Yoshua Bengio. 'Object Recognition with Gradient-Based Learning'. In: ***Shape, Contour and Grouping in Computer Vision***. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 319–345. ISBN: 978-3-540-46805-9. DOI: 10.1007/3-540-46805-6_19. URL: https://doi.org/10.1007/3-540-46805-6_19.

[59] Baopu Li and M.Q.-H. Meng. 'Tumor Recognition in Wireless Capsule Endoscopy Images Using Textural Features and SVM-Based Feature Selection'. In: ***IEEE Trans. Information Technology in Biomedicine*** 16.3 (May 2012), pp. 323–329.

[60] Hui Li, Maryellen L. Giger, Benjamin Q. Huynh and Natasha O. Antropova. 'Deep learning in breast cancer risk assessment: evaluation of convolutional neural networks on a clinical dataset of full-field digital mammograms'. In: ***Journal of Medical Imaging*** 4 (2017), pp. 4–6. DOI: 10.1117/1.JMI.4.4.041304. URL: https://doi.org/10.1117/1.JMI.4.4.041304.

[61] Jun Li, Marion R Nadel, Carolyn F Poppell, Diane M Dwyer, David A Lieberman and Eileen K Steinberger. 'Quality Assessment of Colonoscopy Reporting: Results from a Statewide Cancer Screening Program'. In: ***Diagnostic and Therapeutic Endoscopy*** 2010 (Sept. 2010), p. 419796. ISSN: 1070-3608. DOI: 10.1155/2010/419796. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2948883/.

[62] Rongjian Li, Wenlu Zhang, Heung-Il Suk, Li Wang, Jiang Li, Dinggang Shen and Shuiwang Ji. 'Deep Learning Based Imaging Data Completion for Improved Brain Disease Diagnosis'. In: ***Medical Image Computing and Computer-Assisted Intervention – MICCAI 2014***. LNCS, volu. 2014, pp. 305–312. DOI: 10.1007/978-3-319-10443-0{_}39. URL: http://link.springer.com/10.1007/978-3-319-10443-0_39.

[63] Yibo Li, Liangren Zhang and Zhenming Liu. 'Multi-Objective De Novo Drug Design with Conditional Graph Generative Model'. In: (2018). eprint: arXiv:1801.07299.

[64] David Lieberman, Marion Nadel, Robert A. Smith, Wendy Atkin, Subash B. Duggirala, Robert Fletcher, Seth N. Glick, C. Daniel Johnson, Theodore R. Levin, John B. Pope, Michael B. Potter, David Ransohoff, Douglas Rex, Robert Schoen, Paul Schroy and Sidney Winawer. 'Standardized colonoscopy reporting and data system: report of the Quality Assurance Task Group of the National

Colorectal Cancer Roundtable'. In: ***Gastrointestinal Endoscopy*** 65.6 (2007), pp. 757–766. ISSN: 00165107. DOI: 10.1016/j.gie.2006.12.055.

[65] Moshe Looks, Marcello Herreshoff, DeLesley Hutchins and Peter Norvig. 'Deep Learning with Dynamic Computation Graphs'. In: ***CoRR*** abs/1702.02181 (2017). arXiv: 1702.02181. URL: http://arxiv.org/abs/1702.02181.

[66] L Lundell, J Dent, J Bennett, A Blum, D Armstrong, J Galmiche, F Johnson, M Hongo, J Richter, S Spechler, G Tytgat and L Wallin. 'Endoscopic assessment of oesophagitis: clinical and functional correlates and further validation of the Los Angeles classification'. In: ***Gut*** 45.2 (Aug. 1999), pp. 172–180. ISSN: 0017-5749. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1727604/.

[67] Trygve M. H. Reenskaug. 'Thing-Model-View-Editor – an Example from a planningsystem'. In: (May 1979). Erste Notiz zum Modell-View-Controller-Paradigma mit exemplarischen Beispielen verfasst vom Erfinder persönlich. URL: http://heim.ifi.uio.no/~trygver/1979/mvc-1/1979-05-MVC.pdf.

[68] Hugh D Mai, Robert A Sanowski and J Patrick Waring. 'Improved patient care using the A/S/G/E guidelines on quality assurance: a prospective comparative study'. In: ***Gastrointestinal Endoscopy*** 37.6 (Apr. 2018), pp. 597–599. ISSN: 0016-5107. DOI: 10.1016/S0016-5107(91)70861-4. URL: http://dx.doi.org/10.1016/S0016-5107(91)70861-4.

[69] Maciej A. Mazurowski, Mateusz Buda, Ashirbani Saha and Mustafa R. Bashir. ***Deep learning in radiology: an overview of the concepts and a survey of the state of the art***. 2018. eprint: arXiv:1802.08717.

[70] Thomas M Mitchell. ***Machine Learning***. 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997, pp. 2–3. ISBN: 0070428077.

[71] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra and Martin A. Riedmiller. 'Playing Atari with Deep Reinforcement Learning'. In: ***CoRR*** abs/1312.5602 (2013). arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

[72] Grégoire Montavon, Wojciech Samek and Klaus-Robert Müller. 'Methods for Interpreting and Understanding Deep Neural Networks'. In: ***CoRR*** abs/1706.07979 (2017). arXiv: 1706.07979. URL: http://arxiv.org/abs/1706.07979.

[73] Lena B Palmer, David H Abbott, Natia Hamilton, Dawn Provenzale and Deborah A Fisher. 'Quality of colonoscopy reporting in community practice'. In: ***Gastrointestinal endoscopy*** 72.2 (Aug. 2010), 321–327.e1. ISSN: 0016-5107. DOI: 10.1016/j.gie.2010.03.002. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3087434/.

[74] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga and Adam Lerer. 'Automatic differentiation in PyTorch'. In: (2017).

[75] Konstantin Pogorelov, Olga Ostroukhova, Mattis Jeppsson, Håvard Espeland, Carsten Griwodz, Thomas de Lange, Dag Johansen, Michael Riegler and Pål Halvorsen. 'Deep Learning and Handcrafted Feature Based Approaches for Polyp Detection in Medical Videos'. In: 2018.

[76] Konstantin Pogorelov, Kristin Ranheim, Carsten Griwodz, Thomas de Lange, Sigrun L Eskeland, Dag Johansen, Peter Thelin Schmidt, 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: ***ACM Multimedia Systems***. 2017, pp. 1–6. ISBN: 1234567245. DOI: 10.1145/3083187.3083212.

[77] Konstantin Pogorelov, Michael Riegler, Sigrun Losada Eskeland, Thomas de Lange, Dag Johansen, Carsten Griwodz, Peter Thelin Schmidt and Pål Halvorsen. 'Efficient disease detection in gastrointestinal videos – global features versus neural networks'. In: ***Multimedia Tools and Applications*** 76.21 (Nov. 2017), pp. 22493–22525. ISSN: 1573-7721. DOI: 10.1007/s11042-017-4989-y. URL: https://doi.org/10.1007/s11042-017-4989-y.

[78] Linda Rabeneck, Lawrence F Paszat, Robert J Hilsden, Refik Saskin, Des Leddin, Eva Grunfeld, Elaine Wai, Meredith Goldwasser, Rinku Sutradhar and Therese A Stukel. 'Bleeding and Perforation After Outpatient Colonoscopy and Their Risk Factors in Usual Clinical Practice'. In: ***Gastroenterology*** 135.6 (Dec. 2008), 1899–1906.e1. ISSN: 0016-5085. DOI: 10.1053/j.gastro.2008.08.058. URL: http://dx.doi.org/10.1053/j.gastro.2008.08.058.

[79] Linda Rabeneck, Lawrence F Paszat and Refik Saskin. 'Endoscopist Specialty Is Associated With Incident Colorectal Cancer After a Negative Colonoscopy'. In: ***Clinical Gastroenterology and Hepatology*** 8.3 (Apr. 2018), pp. 275–279. ISSN: 1542-3565. DOI: 10.1016/j.cgh.2009.10.022. URL: http://dx.doi.org/10.1016/j.cgh.2009.10.022.

[80] ***Redux***. 2018. URL: https://redux.js.org/.

[81] Leonard Richardson and Sam Ruby. ***Restful Web Services***. First. O'Reilly, 2007. ISBN: 9780596529260.

[82] Michael Riegler. 'EIR - A Medical Multimedia System for Efficient Computer Aided Diagnosis'. PhD thesis. University of Oslo, 2017, pp. 1–102.

[83] Douglas J Robertson, Laura B Lawrence, Nicholas J Shaheen, John A Baron, Electra Paskett, Nicholas J Petrelli and Robert S Sandler. 'Quality of colonoscopy reporting: a process of care study'. In: ***American Journal Of Gastroenterology*** 97 (Oct. 2002), p. 2651. URL: http://dx.doi.org/10.1111/j.1572-0241.2002.06044.x%20http://10.0.4.87/j.1572-0241.2002.06044.x.

[84] F. Rosenblatt. 'The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain'. In: ***Psychological Review*** (1958), pp. 65–386.

[85] Elisabeth; Rosenthal. ***The $2.7 Trillion Medical Bill***. 2013.

[86] David E Rumelhart, Geoffrey E Hinton and Ronald J Williams. 'Learning representations by back-propagating errors'. In: ***Nature*** 323 (Oct. 1986), p. 533. URL: http://dx.doi.org/10.1038/323533a0%20http://10.0.4.14/323533a0.

[87] G. A. Rummery and M. Niranjan. 'On-Line Q-Learning Using Connectionist Systems'. In: (1994).

[88] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. 'ImageNet Large Scale Visual Recognition Challenge'. In: ***International Journal of Computer Vision (IJCV)*** 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[89] Greta Saino, Luigi Bonavina, John C Lipham, Daniel Dunn and Robert A Ganz. 'Magnetic Sphincter Augmentation for Gastroesophageal Reflux at 5 Years: Final Results of a Pilot Study Show Long-Term Acid Reduction and Symptom Improvement'. In: ***Journal of Laparoendoscopic & Advanced Surgical Techniques. Part A*** 25.10 (Oct. 2015), pp. 787–792. ISSN: 1092-6429. DOI: 10.1089/lap.2015.0394. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4624249/.

[90] Frank Seide and Amit Agarwal. 'CNTK: Microsoft's Open-Source Deep-Learning Toolkit'. In: ***Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining***. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 2135–2135. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2945397. URL: http://doi.acm.org/10.1145/2939672.2945397.

[91] Ramprasaath R. Selvaraju. ***Yes, Deep Networks are great, but are they Trustworthy?*** 2017. URL: https://ramprs.github.io/2017/01/21/Grad-CAM-Making-Off-the-Shelf-Deep-Models-Transparent-through-Visual-Explanations.html.

[92] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh and Dhruv Batra. 'Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization'. In: ***CoRR*** abs/1610.02391 (2016). arXiv: 1610.02391. URL: http://arxiv.org/abs/1610.02391.

[93] L. A. Shafer, J. R. Walker, C. Waldman, C. Yang, V. Michaud, C. N. Bernstein, L. Hathout, J. Park, J. Sisler, G. Restall, K. Wittmeier and H. Singh. 'Factors Associated with Anxiety About Colonoscopy: The Preparation, the Procedure, and the Anticipated Findings'. In: ***Digestive Diseases and Sciences*** 63.3 (Mar. 2018), pp. 610–618. ISSN: 1573-2568. DOI: 10.1007/s10620-018-4912-z. URL: https://doi.org/10.1007/s10620-018-4912-z.

[94] Robyn S Sharma and Peter G Rossos. 'A Review on the Quality of Colonoscopy Reporting'. In: ***Canadian Journal of Gastroenterology and Hepatology*** 2016.i (2016), pp. 1–6. ISSN: 2291-2789. DOI: 10.1155/2016/9423142. URL: 2016Sharma%20http://www.hindawi.com/journals/cjgh/2016/9423142/.

[95] Noam Shussman and Steven D Wexner. 'Colorectal polyps and polyposis syndromes'. In: ***Gastroenterology Report*** 2.1 (Feb. 2014), pp. 1–15. ISSN: 2052-0034. DOI: 10.1093/gastro/got041. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3920990/.

[96] Karen Simonyan, Andrea Vedaldi and Andrew Zisserman. 'Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps'. In: ***CoRR*** abs/1312.6034 (2013). arXiv: 1312.6034. URL: http://arxiv.org/abs/1312.6034.

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

[98] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox and Martin A. Riedmiller. 'Striving for Simplicity: The All Convolutional Net'. In: ***CoRR*** abs/1412.6806 (2014). arXiv: 1412.6806. URL: http://arxiv.org/abs/1412.6806.

[99] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke and Andrew Rabinovich. 'Going Deeper with Convolutions'. In: ***CoRR*** abs/1409.4842 (2014). arXiv: 1409.4842. URL: http://arxiv.org/abs/1409.4842.

[100] Vue Team. ***Vue***. 2018. URL: https://vuejs.org.

[101] ***TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems***. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[102] Mary Than, Jolene Witherspoon, Javed Shami, Prachi Patil and Avanish Saklani. 'Diagnostic miss rate for colorectal cancer: An audit'. In: ***Annals of Gastroenterology*** 28.1 (2015), pp. 94–98. ISSN: 17927463.

[103] Theano Development Team. 'Theano: A Python framework for fast computation of mathematical expressions'. In: ***arXiv e-prints*** abs/1605.02688 (May 2016). URL: http://arxiv.org/abs/1605.02688.

[104] Cedric Van de Bruaene, Danny De Looze and Pieter Hindryckx. 'Small bowel capsule endoscopy: Where are we after almost 15 years of use?' In: ***World Journal of Gastrointestinal Endoscopy*** 7.1 (Jan. 2015), pp. 13–36. ISSN: 1948-5190. DOI: 10.4253/wjge.v7.i1.13. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4295178/.

[105] Rene Vidal, Joan Bruna, Raja Giryes and Stefano Soatto. 'Mathematics of Deep Learning'. In: ***CoRR*** abs/1712.04741 (2017). arXiv: 1712.04741. URL: http://arxiv.org/abs/1712.04741.

[106] Michael B Wallace. 'Endoscopic Removal of Polyps in the Gastrointestinal Tract'. In: ***Gastroenterology & Hepatology*** 13.6 (June 2017), pp. 371–374. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC5495043/.

[107] 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*** 3 (2015), pp. 164–179.

[108] Christopher J. C. H. Watkins and Peter Dayan. 'Technical Note: Q-Learning'. In: ***Mach. Learn.*** 8.3-4 (May 1992), pp. 279–292. ISSN: 0885-6125. DOI: 10.1007/BF00992698. URL: https://doi.org/10.1007/BF00992698.

[109] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs and Hod Lipson. 'Understanding Neural Networks Through Deep Visualization'. In: ***ArXiv e-prints*** (2015). URL: http://arxiv.org/abs/1506.06579.

[110] Matthew D. Zeiler and Rob Fergus. 'Visualizing and Understanding Convolutional Networks arXiv:1311.2901v3 [cs.CV] 28 Nov 2013'. In: ***Computer Vision–ECCV 2014*** 8689 (2014), pp. 818–833. ISSN: 978-3-319-10589-5. DOI: 10.1007/978-3-319-10590-1{_}53. URL: http://link.springer.com/10.1007/978-3-319-10590-1_53%5Cnhttp://arxiv.org/abs/1311.2901%5Cnpapers3://publication/uuid/44feb4b1-873a-4443-8baa-1730ecd16291.

[111] Matthew D Zeiler, Graham W Taylor and Rob Fergus. 'Adaptive deconvolutional networks for mid and high level feature learning'. In: ***Proceedings of the IEEE International Conference on Computer Vision***. ICCV '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 2018–2025. ISBN: 9781457711015. DOI: 10.1109/ICCV.2011.6126474. URL: http://dx.doi.org/10.1109/ICCV.2011.6126474.

[112] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva and Antonio Torralba. 'Learning Deep Features for Discriminative Localization'. In: ***CoRR*** abs/1512.04150 (2015). arXiv: 1512.04150. URL: http://arxiv.org/abs/1512.04150.

[113] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva and Antonio Torralba. 'Object Detectors Emerge in Deep Scene CNNs'. In: ***CoRR*** abs/1412.6856 (2014). arXiv: 1412.6856. URL: http://arxiv.org/abs/1412.6856.

[114] Y.T. Zhou and Rama Chellappa. 'Computation of optical flow using a neural network'. In: (Aug. 1988), 71–78 vol.2.

[115] R. Zhu, R. Zhang and D. Xue. 'Lesion detection of endoscopy images based on convolutional neural network features'. In: ***2015 8th International Congress on Image and Signal Processing (CISP)***. Oct. 2015, pp. 372–376. DOI: 10.1109/CISP.2015.7407907.

Appendices

Appendix A

Source Code

A.1 Mimir Code

The source code for the open-source project *Mimir* is located at https://github.com/stevenah/mimir.

A.2 Training and Evaluation Code

The source code for the training and evaluation scripts used to test our system against is located at https://github.com/stevenah/mimir.

Appendix B

Published Papers

B.1 Paper I — Mimir: An Automatic Reporting and Reasoning System for Deep Learning based Analysis in the Medical Domain

Mimir: An Automatic Reporting and Reasoning System for Deep Learning based Analysis in the Medical Domain

Steven Alexander Hicks
Simula Research Laboratory, Norway
University of Oslo, Norway

Sigrun Eskeland
Department of Medical Research
Bærum Hospital
Vestre Viken Hospital Trust, Norway

Mathias Lux
Klagenfurt University, Austria

Thomas de Lange
Department of Transplantation
Oslo University Hospital, Norway
University of Oslo, Norway

Kristin Ranheim Randel
Cancer Registry of Norway

Mattis Jeppsson
ForzaSys AS, Norway

Konstantin Pogorelov
Simula Research Laboratory, Norway
University of Oslo, Norway

Pål Halvorsen
Simula Metropolitan Center for
Digital Engineering, Norway
University of Oslo, Norway

Michael Riegler
Simula Metropolitan Center for
Digital Engineering, Norway
University of Oslo, Norway

ABSTRACT

Automatic detection of diseases is a growing field of interest, and machine learning in form of deep learning neural networks are frequently explored as a potential tool for the medical video analysis. To both improve the "black box"-understanding and assist in the administrative duties of writing an examination report, we release an automated multimedia reporting software dissecting the neural network to learn the intermediate analysis steps, i.e., we are adding a new level of understanding and explainability by looking into the deep learning algorithms decision processes. The presented open-source software can be used for easy retrieval and reuse of data for automatic report generation, comparisons, teaching and research. As an example, we use live colonoscopy as a use case which is the gold standard examination of the large bowel, commonly performed for clinical and screening purposes. The added information has potentially a large value, and reuse of the data for the automatic reporting may potentially save the doctors large amounts of time.

ACM Reference format:
Steven Alexander Hicks, Sigrun Eskeland, Mathias Lux, Thomas de Lange, Kristin Ranheim Randel, Mattis Jeppsson, Konstantin Pogorelov, Pål Halvorsen, and Michael Riegler. 2018. Mimir: An Automatic Reporting and Reasoning System for Deep Learning based Analysis in the Medical Domain. In *Proceedings of 9th ACM Multimedia Systems Conference, Amsterdam, Netherlands, June 12–15, 2018 (MMSys'18),* 6 pages.
https://doi.org/10.1145/3204949.3208129

Contact author's address: Michael Riegler, Simula Research Laboratory, Oslo, Norway, email: michael@simula.no .

MMSys'18, June 12–15, 2018, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5192-8/18/06…$15.00
https://doi.org/10.1145/3204949.3208129

1 INTRODUCTION

Machine learning has the potential in becoming an important tool in assisting medical professionals to perform medical diagnosis and giving aid in the administrative work that follows. Deep learning has already been shown to work well in various medical fields such as screening for skin cancer, where Esteva et al. [5] (in 2017) presented a deep convolutional neural network (CNN) with the ability to diagnose skin cancer at the level of a trained dermatologist. This shows that deep learning can successfully be applied to fields of medical expertise, but experts are still left with the work of documenting the procedure through written reports. With the amount of data gathered through medical examinations rapidly increasing, we need a way to process this information without drowning clinicians in administrative work.

One solution to this problem is through automatic methods, e.g. deep learning, where the collected data is automatically compiled into summaries, conveying key aspects from the medical procedure. This would not only relieve doctors from parts of the administrative process, but could be used as a teaching tool for medical students. Through multimedia enriched reports, medical doctors in training can learn based on real data according to case-based teaching and problem-based learning strategies. Thus, multimedia summarization for automated report generation is a much needed feature [25].

A major obstacle with using complex automatic methods is that the inner workings are often hard to understand, making it difficult to determine how and why it produces its results, i.e., deep learning is often used as a "black box". This is especially problematic in the field of medicine among others, where the doctors need to justify a decision besides referring to the system itself. To the best of our knowledge, this is yet an unexplored area of research. Moreover, no open-source software exists that could support researchers in both domains, computer science and medicine, to perform much needed research in this direction.

In an effort to open this "black box" and assist in the documentation of medical examinations, we present *Mimir*, an automated

multimedia reporting system, which goes beyond the creation of medical reports by adding a level of understanding and explainability through methods of looking into a deep neural networks decision process. The presented open-source software can be used for easy retrieval and reuse of data for automatic report generation, comparisons, teaching and research. As a first use-case, we use live colonoscopy, which is the conventional (and gold standard) method of screening the large intestine. Through insertion of a long flexible tube equipped with a tiny camera into the anus, it allows for direct inspection of the bowel mucosa. This plays an essential role in the diagnosis of various abnormalities commonly found in the lower gastrointestinal (GI) tract, such as inflammation, colorectal cancer and its precursors (polyps). Before starting *Mimir*, we developed a live detection system [26, 27] which analyses a direct video stream from a colonoscopy, and gives live visual feedback whether or not anything is detected [28]. This however, does not explain why the system signaled a detection and does not provide any form of text summaries of the overall examination process.

We aim not to just create reports containing text and most representative multimedia content such as images or videos, but also to explain to the users why a certain image has been identified as relevant. The main contribution therefore is to provide researchers and domain experts a novel way of using intermediate visual representations of deep neural network layers and results to increase understanding, trust and usefulness. The representations created by the system can be used for example in disease detection scenarios.

Below, we briefly describe the system based on Google's TensorFlow, give a brief introduction to the code and installation, and discuss some examples of how to use *Mimir*.

2 RELATED WORK

Over the last few years, deep learning has proved to be a powerful tool in many fields and is now (2018) considered the gold standard in many areas such as language translation, object recognition and image captioning [17]. However, generation of quality medical reports goes beyond transforming explicit information from one media to another. It often involves multiple different forms of media, which must be combined in order to argue and justify the diagnosis of a medical expert.

The current practice of reporting medical procedures is an essential, yet cumbersome, part of a clinicians' daily work. Research shows that approximately one-sixth of U.S. physicians working time is spent on administrative tasks, taking time away from direct-patient care and lessening job satisfaction [35].

In addition, within GI endoscopy, there is a general lack of language standardization, which may result in poor communication between health care providers. Thus, following a systematic approach to document the findings of an endoscopic procedure would be favorable in an attempt to achieve a certain level of consistency within GI reporting. An automated reporting system based on automatic video analysis would be extremely helpful in this regard, and help contribute to the implementation of the *Minimal Standard Terminology* (MST) recommended by the World Endoscopy Organization (WEO). Additionally, the standardization of medical reporting related to endoscopic procedures is listed as a requirement by the European Society of Gastrointestinal Endoscopy (ESGE) [4].

In the field of medicine, data driven methods can be questionable if the results are not reproducible or comprehensible by the medical experts using them. With deep learning in particular, the results of automatic recognition are extremely helpful, but we are still unable to fully understand the rationale behind the decisions made by the algorithm. This has lead to a trade-off between more comprehensible models and models that yield a higher accuracy, where simpler models are often chosen over those with higher accuracy as they are typically easier to interpret. Recent developments have provided theoretical and visual approaches to better understand the decisions made by a deep neural network. Theoretical approaches rely on describing the underlying mathematics, taking a closer look at how the individual mathematical properties produce a given result [18, 34]. This is useful, but interpreting such descriptions require a deep understanding of the math and technology of deep learning, something we cannot expect end-users to have. Visual approaches try to present layers using a variety of visualization techniques such as saliency maps or other forms of visual representations (texture maps, heat maps, etc.) [29, 37] and come closer to gaining a better understanding of the classification process without detailed technical knowledge of the underlying system.

It is worth noting that medical doctors indicated that a tool for automatic text generation was not that important to them. It was more important for them to understand the underlying analysis process, and receive support in generating high quality documents through consistent means [25]. *Mimir* aims to meet them halfway. By including the doctors in the analysis process, we give them an intuitive way to understand how and why the system produces its results.

In sum, the goal is to create a tool that aids in the production of a structured and semantically correct reports, composed of text and images taken from a medical procedure (GI endoscopy in our case). Moreover, the tool needs to make the process understandable and reproducible for non-technical users to ensure the trust of the doctors and patients involved.

A recent approach [15] investigates the possibility of creating reports from x-ray images employing neural image captioning methods [36]. A network is trained from a dataset of images along with the reports. Closest to our approach is the work described in [38], where microscope images are fed to a neural network to generate reports and retrieve relevant images of symptoms in addition to visualization of the attention of the network to support the rationale of the decision made by the network. Both approaches focus on images already classified as relevant by being part of a diagnostic process, whereas the second paper adds the dimension of the rationale of the generated report.

3 SYSTEM DESCRIPTION

Mimir can be described as a framework with three main functionalities:

- The system was designed to aid medical doctors in making informed decisions regarding diagnosis of diseases found during examinations, such as diagnosis of disease found in the GI tract during a colonoscopy.
- *Mimir* creates automatic reports based on the automatic analysis of images and videos and reduces the time spent

on the administrative tasks that follow an endoscopic examination, e.g., documentation by written reports. This is shown in figure 1 where the doctor uses the system to understand the analysis done by the neural network, and use this information to reach a diagnosis and generate the accompanied report.
- *Mimir* can be used by researchers and engineers designing deep learning architectures such as CNNs to gain a better understanding of the evaluation and reactions of their models, e.g., by understanding which parts of an image confuse the algorithm and if additional pre-processing steps are needed.

In *Mimir*, we use a deep CNN to analyse image or video data to perform different classification task, e.g., automatic detection of diseases. This process is made transparent to the users through a tool that dissects the individual layers of a CNN, making it possible to see the basis for the decisions made by the system and on what regions of an image the algorithm activates for a target class. This is a critical piece in building trust among users of the system like medical professionals who need to rely on the systems output without the technical knowledge of the internal workings of a CNN. Additionally, it allows for discovering fallacies within the model itself and the dataset used to train it.

Using the guided grad-CAM technique [30], we generate visual representations of an image as it moves through the network, showing what regions of the image correspond to a target class at the point of a selected layer. The process is shown in figure 4, and starts with the user selecting an input image, target layer and target class using the web-interface (Shown at the bottom of figure 4). Based on the selection, the system generates three visualizations of the image (the three visualizations are shown in figure 2 together with the original image). Figure 2a is the original image before any processing. Figure 2b is a grad-CAM (A generalization of class activation map (CAM) [39]) representation of the image which shows what regions of the image correspond the the selected target class. Figure 2c is the saliency map generated using guided back-propagation, this shows the positive activations of the target layer, and is not class specific. Figure 2d depicts the guided grad-CAM representation of the image, which is a combination of the grad-CAM and saliency map. From the three visualizations, the system presents the grad-CAM and the guided grad-CAM to the user.

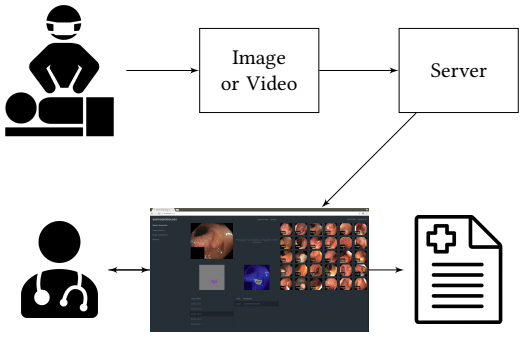


Figure 1: Reporting system workflow. Images and videos are collected and analyzed during the examination. After the examination, the system can give intermediate insights from the neural network for the presented findings and a modifiable report draft is presented in order to produce a final report including text, representative images and video clips.

Figure 3 shows five guided grad-CAM representations of an image containing a polyp using polyp as the target class, each corresponding to the last convolutional layer in the five convolutional blocks of a VGG-19 CNN. The written reports are produced through a "what you see is what you get" (WYSIWYG) editor, with additional options for image attachments.

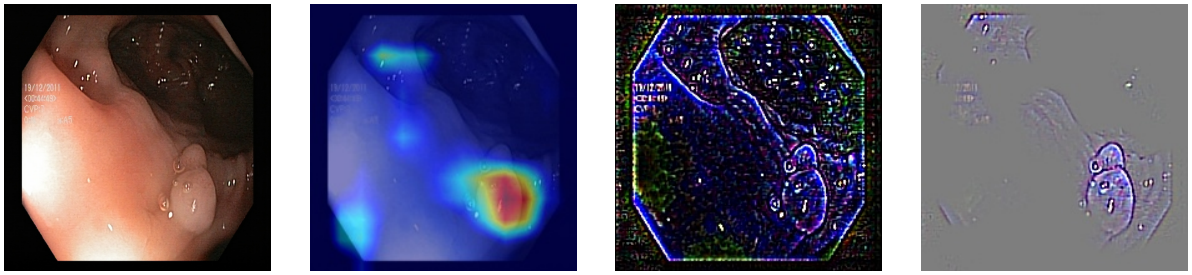
4 CODE

The code repository [14] contains the server and web application, clearly separated in their respective directories. The web application uses a standard flux architecture [6], implemented using React [8] and Redux [1]. The code is fully documented and tested using the testing framework Jest [7]. The advantage of making a client web application is the ease of access and portability of being available on any device that supports a web browser. The server is written in Python (using the micro-framework Flask[10]), and is accessible through a RESTful [24] API, with endpoints for interaction with the underlying deep neural network. As mentioned previously, the image/frame analysis is done using deep learning, specifically a deep CNN. The CNN uses a standard VGG-19 architecture [31] trained on the Kvasir version 2 dataset [22] and is implemented using the Keras deep learning framework [16] with a Tensorflow backend [2]. *Mimir* is licensed under the terms of the GNU General Public License (GPL) version 3, as published by the Free Software Foundation and available on Github [14].

The image/frame visualizations are done using the guided grad-CAM approach [30], and is generated on the fly once the user selects an image, target layer and target class to visualize. A target layer and target class can be selected individually from their respective lists (as seen in the web-interface of figure 4). The layer selection list contains each convolutional layer in the underlying CNN, and the class selection list contains each class supported by the system. The guided grad-CAM technique combines the class discriminative properties of a CAM with pixel level detail of guided back-propagation saliency maps [32]. Our implementation is based on the Selvaraju et al. paper [30] and is implemented using Keras backend functions. The current system uses two image representations to explain the CNN, grad-CAM and guided grad-CAM. The overall process of generating these two visualizations can be broken into three parts;

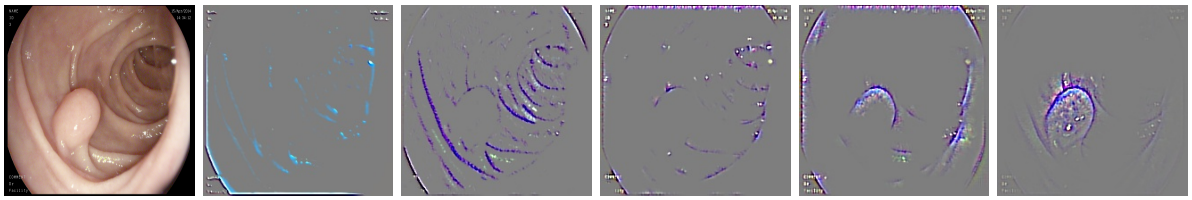
(1) Generate a grad-CAM representation using the a target layer and target class with respect to the input image.
(2) Generate a guided back-propagation saliency map using the same target layer as used when generating the grad-CAM with respect to the input image.
(3) Combine the grad-CAM with the saliency map made in the previous two steps to produce the guided grad-CAM visualization.

Evidently, the grad-CAM is an intermediate step of the guided grad-CAM process, so both representations are created through the



(a) Original Image (b) Grad-CAM (c) Guided Backprop (d) Guided Grad-CAM

Figure 2: Image representations used by the reporting system to explain decisions.



(a) Original (b) Block 1 (c) Block 2 (d) Block 3 (e) Block 4 (f) Block 5

Figure 3: Guided grad-CAM representation of an image at the last convolutional layer of each convolutional block.

same process. A visual representation of the process can be seen in figure 4.

The visualization process starts once the user has selected an image, layer and class for further inspection. With an image, target layer and target class selected, we calculate the gradient of the target layer using the loss of the target class in regards to the image. These gradients are globally average pooled to get the weights, which is multiplied with output of the target layer and passed through a relu function to produce the grad-CAM. The grad-CAM is re-sized back to the dimensions of the original image and its values squashed between 0 and 1 before a blue-red heat map filter is applied.

In order to generate the guided back-propagation saliency map, we start by replacing the activations of the original network with a slightly modified relu function. During back-propagation, a traditional relu would let all gradients whose inputs where larger than 0 pass. We change this by adding an additional rule which discards all gradients that have value below 0 (i.e. negative gradients), thereby only back-propagating the positive influence on the activations. With this modified network we calculate the gradients of the target layer with respect to the input image, these gradients represent our saliency map.

Once the grad-CAM (Figure 2b) and saliency map (Figure 2c) have been computed, we simply multiply them together to produce the guided grad-CAM (Figure 2d) representation. This together with the grad-CAM is used in our system.

5 INSTALLATION

As mentioned in section 4, the system is built using the micro-framework Flask, which includes a built-in development sever, making it easy to start a local instance of the application. Note that the development server is not meant to be deployed to a production environment. For a production environment we recommend deployment using a popular web server such as Nginx [19] or Apache HTTP Server [3], or by using the pre-built Docker Image available through Docker hub [13]. There are two primary ways of getting the system up and running, pulling the git repository from Github [14] or pulling the pre-configured docker image from Docker hub [13].

Setting up the system using the git repository requires multiple steps of pre-configuration before we can launch the local development server. This includes;

(1) Setup a Python 3.6 run-time environment with the necessary dependencies.
(2) Configure Keras (2.0.8) [16] to use Tensorflow [11] as a backend.
(3) Install OpenCV [33] with FFmpeg [9] support.
(4) Install CuDNN [21] and CUDA toolkit [20] for GPU support (this step is optional, but highly recommended).

A more detailed setup and configuration guide can be found in the applications Github repository. With the environment setup, we can launch a local development server by running `app.py` using Python 3.

For an easier setup, we recommend using the pre-built Docker image available at Docker hub [13]. The container includes a pre-configured Python environment with all the necessary dependencies installed, CUDA 8 and cuDNN 6 for Nvidia GPU support, and served using an Ngnix server instance.

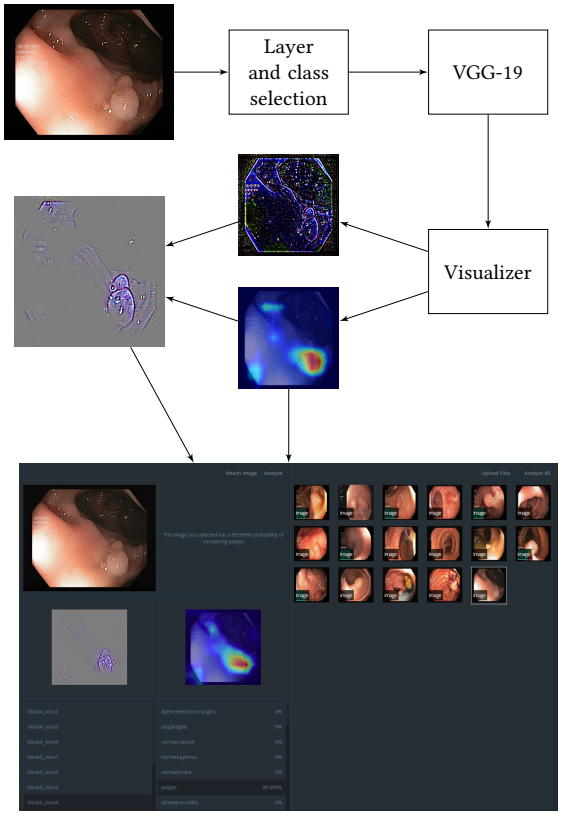


Figure 4: An overview of how we produce the two visualizations included in the image analysis, and how it is presented in the user interface where a visualization of the different convolutional blocks can be selected.

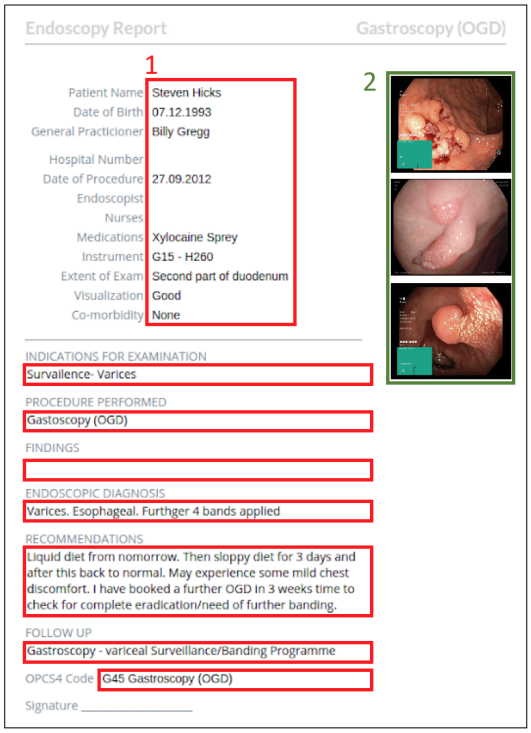


Figure 5: An example of an automatic generated report. The red area marked (1) shows the editable text fields. The green area (2) shows the images chosen for the report. Report based on sample taken from Wrestling the Octopus [12].

6 USAGE

The web application can be accessed using the configured host IP and port. The following examples will describe typical scenarios we imagine this tool being used for.

6.1 Example Scenario A - Verify the Prediction of a Diagnosis

After getting the diagnosis based on the analysis of the colonoscopy examination video, we would like to verify that the network does in fact detect the diagnosed abnormality presented. After the examination, the frames where abnormalities are detected are automatically presented to the user on the image analysis web-page. For a given frame, the user can look through the network and verify that the network does in fact detect the abnormality related to the diagnosis. An example is shown in figure 2 where we clearly see that the network detects the polyp located in the lower right corner of the image. Note that not all detections are this obvious, and the additional image representations are thus even more useful when abnormalities are difficult to detect.

6.2 Example Scenario B - Generating a Colonoscopy Report

After a colonoscopy, the video produced is automatically passed through the system and analysed for abnormalities. Based on the diagnosis, the system would present images that support the diagnoses (which can be further examined as described above in section 6.1). This would save the user time by not having to screen the frames of the video for the diagnosed abnormality and manually select image candidates.

The report generation tool provides basic text editing through a WYSIWYG interface, with additional options for adding images to support the findings described in the report. The tool presents a live preview of the printed document, which may be modified by clicking the various text fields of the report. Images may be manually or automatically added through image uploads or by taking images already part of the system. Note that the current format of the report is taken from Wrestling the Octopus [12], and is used just as an example. In a real world use-case, the format of the report would be tailored to the needs of the institution. An

example report can be found in Figure 5 with text and pre-selected images (that the user can change).

7 CONCLUSION

Nowadays, neural networks are widely used, but there is still a lack of understanding when it comes to how they operate and on what their output is based on, even more so among non-technical users. This is may be sufficient for many fields, but in mission-critical areas such as medicine (among others), the clinicians often need to understand why a particular marking is detected. To improve the understanding of the internal decision process of a deep neural network, and to build trust among its users, we made the source code of our system publicly available. *Mimir* allows for dissecting of deep neural networks, enabling investigation and understanding of the networks layers and outputs. Our system can also use this information to create automatic reports from the analysis of images or videos. In this paper, we have briefly described the system based on Google's TensorFlow, given a introduction to the code and installation, and discussed some examples of how to use *Mimir*.

REFERENCES

[1] 2018. Redux. (2018). https://redux.js.org/

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, and others. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[3] Apache. 2018. Apache HTTP Server Project. (2018). https://httpd.apache.org/

[4] Michael Bretthauer, Lars Aabakken, Evelien Dekker, Michal F Kaminski, Thomas Rösch, Rolf Hultcrantz, Stepan Suchanek, Rodrigo Jover, Ernst J Kuipers, Raf Bisschops, and others. 2016. Reporting systems in gastrointestinal endoscopy: Requirements and standards facilitating quality improvement: European Society of Gastrointestinal Endoscopy position statement. *United European gastroenterology journal* 4, 2 (2016), 172–176.

[5] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. 2017. Dermatologist-level classification of skin cancer with deep neural networks. *Nature* 542, 7639 (feb 2017), 115–118. https://doi.org/10.1038/nature21056

[6] Facebook. 2018. Flux. (2018). https://facebook.github.io/flux/

[7] Facebook. 2018. Jest. (2018). https://facebook.github.io/jest/

[8] Facebook. 2018. React. (2018). https://reactjs.org/

[9] FFmpeg. 2018. FFmpeg. (2018). https://www.ffmpeg.org/

[10] Flask. 2018. Flask. (2018). http://flask.pocoo.org/

[11] Google. 2018. Tensorflow. (2018). https://www.tensorflow.org/

[12] Nigel H. 2015. The Crohnoid Blog. (2015). http://www.wrestlingtheoctopus.com/the-a-to-z-of-my-crohns/

[13] Steven Hicks. 2018. Mimir Docker Repository. (2018). https://hub.docker.com/r/stevenah/mimir/

[14] Steven Hicks. 2018. Mimir Github Repository. (2018). https://github.com/Stevenah/mimir

[15] Baoyu Jing, Pengtao Xie, and Eric Xing. 2017. On the Automatic Generation of Medical Imaging Reports. *arXiv preprint arXiv:1711.08195* (2017).

[16] Keras. 2018. Keras: The Python Deep Learning library. (2018). https://keras.io/

[17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[18] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2017. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing* (2017).

[19] NGINX. 2018. NGINX. (2018). https://nginx.org/en/

[20] Nvidia. 2018. Nvidia CUDA Toolkit. (2018). https://developer.nvidia.com/cuda-toolkit

[21] Nvidia. 2018. Nvidia CuDNN. (2018). https://developer.nvidia.com/cudnn

[22] Konstantin Pogorelov, Kristin Ranheim Randel, Carsten Griwodz, Sigrun Losada Eskeland, Thomas de Lange, Dag Johansen, Concetto Spampinato, Duc-Tien Dang-Nguyen, Mathias Lux, Peter Thelin Schmidt, Michael Riegler, and Pål Halvorsen. 2017. KVASIR: A Multi-Class Image Dataset for Computer Aided Gastrointestinal Disease Detection. In *Proc. of MMSYS*. 164–169. https://doi.org/10.1145/3083187.3083212

[23] Konstantin Pogorelov, Michael Riegler, Sigrun Losada Eskeland, Thomas de Lange, Dag Johansen, Carsten Griwodz, Peter Thelin Schmidt, and Pål Halvorsen. 2017. Efficient disease detection in gastrointestinal videos – global features versus neural networks. *Multimedia Tools and Applications* 76, 21 (01 Nov 2017), 22493–22525. https://doi.org/10.1007/s11042-017-4989-y

[24] Leonard Richardson and Sam Ruby. 2007. *Restful Web Services* (first ed.). O'Reilly.

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

[26] Michael Riegler, Konstantin Pogorelov, Pål Halvorsen, Thomas de Lange, Carsten Griwodz, Peter Thelin Schmidt, Sigrun L. Eskeland, and Dag Johansen. 2016. EIR - Efficient Computer Aided Diagnosis Framework for Gastrointestinal Endoscopies. In *Proc. of CBMI*.

[27] 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. 2016. Computer Aided Disease Detection System for Gastrointestinal Examinations. In *Proc. of MMSys*.

[28] 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. 2016. Computer Aided Disease Detection System for Gastrointestinal Examinations. In *Proc. of MMSYS*. 29:1–29:4. https://doi.org/10.1145/2910017.2910629

[29] Christin Seifert, Aisha Aamir, Aparna Balagopalan, Dhruv Jain, Abhinav Sharma, Sebastian Grottel, and Stefan Gumhold. 2017. Visualizations of Deep Neural Networks in Computer Vision: A Survey. In *Transparent Data Mining for Big and Small Data*. Springer, 123–144.

[30] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. 2016. Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization. *CoRR* abs/1610.02391 (2016). arXiv:1610.02391 http://arxiv.org/abs/1610.02391

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

[32] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. 2014. Striving for Simplicity: The All Convolutional Net. *CoRR* abs/1412.6806 (2014). arXiv:1412.6806 http://arxiv.org/abs/1412.6806

[33] OpenCV team. 2018. Open Source Computer Vision Library (OpenCV). (2018). https://opencv.org/

[34] Rene Vidal, Joan Bruna, Raja Giryes, and Stefano Soatto. 2017. Mathematics of Deep Learning. *arXiv preprint arXiv:1712.04741* (2017).

[35] Steffie Woolhandler and David U Himmelstein. 2014. Administrative Work Consumes One-Sixth of U.S. Physicians' Working Hours and Lowers Their Career Satisfaction. 44 (10 2014), 635–42.

[36] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proc. of ML*. 2048–2057.

[37] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. 2015. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579* (2015).

[38] Zizhao Zhang, Yuanpu Xie, Fuyong Xing, Mason McGough, and Lin Yang. 2017. Mdnet: A semantically and visually interpretable medical image diagnosis network. In *Proc. of IEEE CVPR*. 6428–6436.

[39] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. 2015. Learning Deep Features for Discriminative Localization. *CoRR* abs/1512.04150 (2015). arXiv:1512.04150 http://arxiv.org/abs/1512.04150

B.2 Paper II — Comprehensible Reasoning and Automated Reporting of Medical Examinations Based on Deep Learning Analysis

Comprehensible Reasoning and Automated Reporting of Medical Examinations Based on Deep Learning Analysis

Steven Alexander Hicks
Simula Research Laboratory, Norway
University of Oslo, Norway

Konstantin Pogorelov
Simula Research Laboratory, Norway
University of Oslo, Norway

Mathias Lux
University of Klagenfurt, Austria

Mattis Jeppsson
ForzaSys AS, Norway

Kristin Ranheim Randel
Cancer Registry of Norway

Thomas de Lange
Department of Transplantation
Oslo University Hospital, Norway
University of Oslo, Norway

Sigrun Eskeland
Department of Medical Research
Bærum Hospital
Vestre Viken Hospital Trust, Norway

Pål Halvorsen
Simula Metropolitan Center for
Digital Engineering, Norway
University of Oslo, Norway

Michael Riegler
Simula Metropolitan Center for
Digital Engineering, Norway
University of Oslo, Norway

ABSTRACT

In the future, medical doctors will to an increasing degree be assisted by deep learning neural networks for disease detection during examinations of patients. In order to make qualified decisions, the black box of deep learning must be opened to increase the understanding of the reasoning behind the decision of the machine learning system. Furthermore, preparing reports after the examinations is a significant part of a doctors work-day, but if we already have a system dissecting the neural network for understanding, the same tool can be used for automatic report generation. In this demo, we describe a system that analyses medical videos from the gastrointestinal tract. Our system dissects the Tensorflow-based neural network to provide insights into the analysis and uses the resulting classification and rationale behind the classification to automatically generate an examination report for the patient's medical journal.

ACM Reference format:
Steven Alexander Hicks, Konstantin Pogorelov, Mathias Lux, Mattis Jeppsson, Kristin Ranheim Randel, Thomas de Lange, Sigrun Eskeland, Pål Halvorsen, and Michael Riegler. 2018. Comprehensible Reasoning and Automated Reporting of Medical Examinations Based on Deep Learning Analysis. In *Proceedings of 9th ACM Multimedia Systems Conference, Amsterdam, Netherlands, June 12–15, 2018 (MMSys'18),* 4 pages.
https://doi.org/10.1145/3204949.3208113

1 INTRODUCTION

Machine learning has shown much potential in becoming an important asset to medical doctors performing disease detection during patient examinations. As a result of this, we may see a decrease in diagnostic errors (in the form of missed disease), increase in number of patients, and further improve the quality of medical care. Additionally, a significant part of a medical professional's time is spent preparing reports after the performed procedures. Multimedia research can significantly support this phase by collecting patient and examination data and providing automatically generated summaries conveying key information of the performed procedures, e.g., video frames with detected objects. An automatically generated report is also useful for training medical experts: through multimedia enriched reports, medical doctors in training can learn based on real data according to case-based teaching and problem-based learning strategies. Thus, multimedia summarization for automated report generation is a much needed feature [20], but it is still in its infancy. One major obstacle is that it is not always comprehensible or reproducible why an automatic detection system marks a finding, i.e., the machine learning system is a black box. In the field of medicine, among others, this is not acceptable as medical doctors often need the underlying rationale behind a decision besides the decision from the system itself. To the best of our knowledge, this is yet an unexplored area of research.

To both improve the "black box"-understanding and assist the examination reporting, we research automated multimedia summarization methods with a semantic nature exploiting domain ontologies. Based on the detection system, the video backend may be used for easy retrieval and reuse of data for automatic report generation, comparisons, teaching and research. As a case study, we use live colonoscopy. This is the gold standard examination of the large bowel, commonly performed for clinical and screening purposes. It allows inspection of the bowel mucosa, essential for the diagnosis of abnormalities such as inflammation, colorectal cancer and its precursors (polyps). We have previously developed a live detection system [21, 22]. Under a colonoscopy, the system analyses the captured video frames and gives visual feedback to the doctor if something abnormal is detected [23]. In this paper, we demonstrate how this system can be extended to colonoscopy documentation. After the colonoscopy, an overview (Figure 1) is given where the doctors can make changes or corrections, and add

Contact author's address: Michael Riegler, Simula Research Laboratory, Oslo, Norway, email: michael@simula.no .

MMSys'18, June 12–15, 2018, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5192-8/18/06…$15.00
https://doi.org/10.1145/3204949.3208113

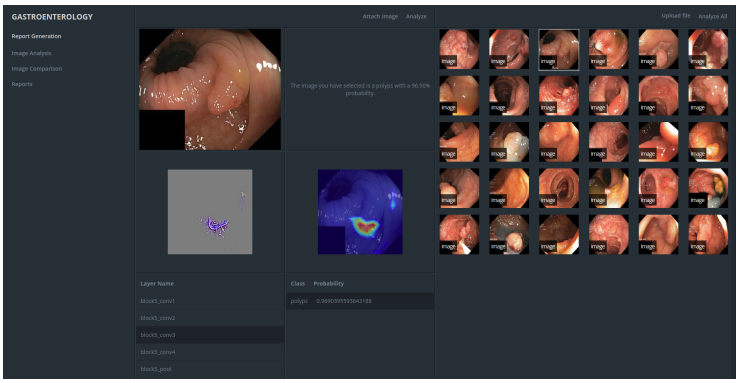


Figure 1: Report and feedback interface, where you may browse through the different neural network layers.

additional information. This can then be stored for later purposes or used in a written endoscopy report. Further, it can be practical to store high quality images of the most important parts [5], i.e., our reporting system also recommends images (frames) to be included in the report and dissects the neural network to give a reasoning why the image is selected.

2 MEDICAL AUTOMATIC REPORTING

Deep learning has greatly improved automatic methods for speech to text conversion, object recognition and image captioning [13]. Structured reporting for colonoscopy procedures, however, is beyond transforming explicit information from one media to another. It also involves finding relevant pieces from multiple modalities and putting them together into a readable report, that supports and argues the diagnosis of a medical expert. In the field of medicine, written reporting of medical procedures is an essential, but cumbersome, part of the physicians' daily work, and the quality and completeness of the reports are critical to the patients care and well being. A more automated reporting system based on automatic video analysis would be extremely helpful for medical experts and contribute to a standardization of the medical report and the implementation of the *Minimal Standard Terminology* (MST) recommended by the World Endoscopy Organization (WEO). Also, the European Society of Gastrointestinal Endoscopy lists the standardization of medical reporting in endoscopic procedures as a requirement [4].

In mission-critical domains, such as the medicine, data driven methods can be questionable if the results are not reproducible or comprehensible by experts within their field. With deep learning in particular, the results of automatic recognition are extremely helpful, but we are still not able to fully understand the rationale of every decision of a network. Theoretical approaches to explain the decisions of a deep neural network have been discussed [14, 29], but it is important to address the problem of understanding and trust among non-technical users, i.e., medical experts and doctors. More visual approaches that present layers using tools such as heat maps or visual representations (texture, heat maps, etc.) [24, 31] come closer to what users can grasp without detailed technical knowledge. All in all, the goal is a tool that generates a structured and readable report composed of text and images from a medical procedure. Moreover, the tool has to this understandable and reproducible for non-technical users to ensure the trust of doctors and patients involved. A recent approach [11] investigates the possibility of creating reports from x-ray images employing neural image captioning methods [30]. A network is trained from a dataset of images along with the reports. Closest to our approach is the work described in [32], where microscope images are fed through a neural network to generate reports and retrieve relevant images of symptoms in addition to an attention map to support the rationale of the networks decision. Both approaches focus on images already classified as relevant by being part of a diagnostic process, whereas the second paper adds the dimension of the rationale of the generated report.

Medical doctors indicated that generating automatic text is not the most important feature for them. More importantly, they need to understand the decisions of the algorithms in an easy and intuitive way, and at the same time, receive support for generating high quality, structured reports [20].

3 ARCHITECTURE AND IMPLEMENTATION

The objectives of our system is to increases classification understanding and reduce the time spent on administrative tasks related to a colonoscopy, e.g., documentation by written reports. The system reports abnormalities commonly found in the gastrointestinal (GI) tract, such as polyps and esophagitis, based on analysis of frame data taken from a video stream. The frames with detected abnormalities are presented to the user for further analysis, with the most prominent images (highest probability of abnormality detected) suggested as attachments to be included in the written report. It is important that the process of disease detection is transparent, i.e., by being able to comprehend why the system concluded as it did and on what basis the diagnosis was set. This is a key component in building trust among the medical professionals who rely on the system to make qualified medical decisions. In addition to building truest among our expected users, it allows us to detect weaknesses in the tool itself and the dataset used to train it. Insight into the analysis process is done using various visualization techniques to generate intermediate representations of an image as it moves through a neural network, specifically as it moves through the convolutional layers of a convolutional neural network (CNN).

This gives us a peek into the decision process of the neural network, showing what regions in the image correspond to a given prediction. This process will be discussed further in section 3.1.

The system is accessed through a React [6] based web application, backed up by a RESTful server API written in Python (using the micro-framework Flask [8]). Image analysis is done using a CNN, specifically a standard VGG-19 model [26] trained on the Kvasir version 2 dataset [18] and is implemented using the Keras deep learning framework [12] with a Tensorflow backend [2]. Figure 2 shows the typical case in how we imagine this tool being used for visualizing the analysis process of passing images/frames through the CNN and report generation based on the results from the analysis.

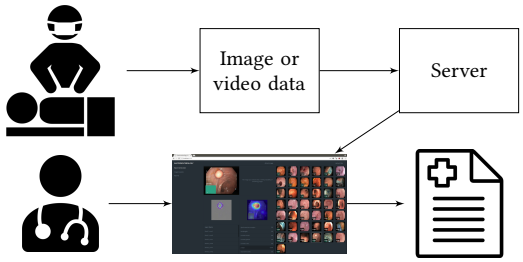


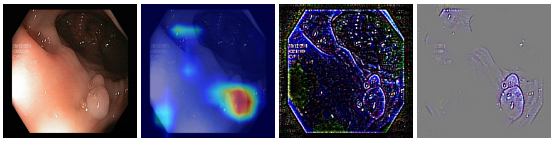
Figure 2: The expected work flow of the reporting system. Images and videos are collected and analyzed during the examination. After the examination, a modifiable report draft is presented to the medical expert in order to produce a final report including text, representative images and video clips.

3.1 Image/Frame Visualization

The visualizations process works on images and videos, with videos being split into frames and processed individually in the same way as a single image. The neural network image representations are generated using a guided grad-cam approach [25], which combines the pixel-level detail of guided back-propagation saliency maps [33] with the class discriminative properties of class activation maps (CAMs) [27]. The result is a high quality image with class discrimination on a pixel level. Each image representation is done with respect to a target class and layer, making it possible to look back through the network and see what less abstract features were picked up by the network.

Figure 3 shows the original image (Figure 3a) and three additional presentations generated by the tool. Figure 3b (grad-CAM) shows the the class-specific regions of the image with respect to a target class at a given layer of the network. Figure 3c (guided back-propagation saliency map) shows a pixel-level representation of what the network sees at a given layer. Figure 3d is a combination of the first two representations, combining the pixel-level detail of the saliency map with the class discriminative features of the grad-CAM.

We start the visualization process by selecting a target layer and class we wish to visualize for a given image. We calculate the gradient of the target layer using the loss for the target class in regards to the input image. The gradients are globally average pooled and multiplied with the output of the target layer. The result is passed through a relu function before it is re-sized back to the



(a) Original Image **(b)** Grad-CAM **(c)** Guided Backprop **(d)** Guided Grad-CAM

Figure 3: Image representations used to explain decisions.

dimensions of the original image. Finally, we squash the values between 0 and 1, and apply a red-blue heatmap filter.

To generate the guided back-propagation saliency map, we start by replacing the activations of our original network with a modified relu activation. During back-propagation, a traditional relu activation would let all gradients whose inputs where larger than 0 pass. We change relu by adding the additional rule of discarding all gradients that are below 0, thereby only back-propagating the positive influence on the activations. With this modified network we calculate the gradients of the target layer with respect to the input image which gives us the saliency map.

Once the grad-CAM and saliency map have been computed, we simply multiply them together to get the guided grad-CAM visualization. This together with the grad-CAM is used in our tool.

3.2 Report Generation

The current state of report generation provides basic functionalities such as changing text and adding additional images. The system presents a preview of the printed report to the users, with direct modification available by clicking and editing the various reports. Images from the analysis can be manually or automatically added or removed. An example report can be found in Figure 4 with text and pre-selected images that the user can change.

4 SETUP AND USAGE

The system is built using the micro-framework Flask [8], which includes a built-in development sever, making it easy to start a local instance of the system. Note that this is not meant to be deployed to a production environment. For a production environment, we can either deploy the application using a popular web server such as Nginx [15] or Apache HTTP Server [3], or by using the pre-built Docker Image available through Docker hub [9]. Setting up the system using the git repository requires several steps of pre-configuration before we can launch the local development server. This includes setting up a Python 3.6 run-time environment and installing the necessary Python dependencies, installing OpenCV [28] with FFmpeg [7] support, configuring Tensorflow and Keras, and optionally (but highly recommended) configuring cuDNN [17] and CUDA [16] for GPU support. With the environment setup, we can launch a local development server by running `app.py` using Python 3. A more detailed setup and configuration guide can be viewed at the application's Github repository [10], but for an easier setup, we recommend using the pre-built Docker image available at Docker hub[9]. The image includes a pre-configured Python environment with the necessary dependencies, CUDA 8 and cuDNN 6 for Nvidia GPU support, and hosted using Nginx. Once the tool is up and running, it can be accessed through a web browser using the configured host IP and port.

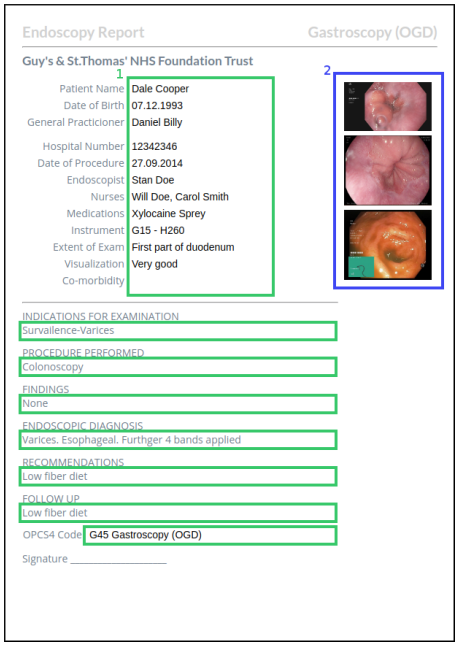


Figure 4: An example of an automatic generated report. The green area marked (1) shows the editable text fields. The blue area (2) shows the images chosen for the report. Report based on sample taken from Wrestling the Octopus [1].

5 DEMO

In the proposed demo, the participants will be able to see how the system works in real time. In particular, video(s) with disease will be available, and the participants may run it through the system. After the deep learning neural network has analysed the video frames, a screen as shown in Figure 1 will be displayed showing the results of the analysis for each layer. From the list of images selected by the system, the user can select one and see how it has been processed through the network by showing images of the intermediate representations and saliency maps (or heatmaps). Finally, a report can be automatically generated from this interface including both text and images (frames). This report can also be modified after it is created.

REFERENCES

[1] 2015. The Crohnoid Blog. (2015). http://www.wrestlingtheoctopus.com/the-a-to-z-of-my-crohns/

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, and others. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[3] Apache. 2018. Apache HTTP Server Project. (2018). https://httpd.apache.org/

[4] Michael Bretthauer, Lars Aabakken, Evelien Dekker, Michal F Kaminski, Thomas Rösch, Rolf Hultcrantz, Stepan Suchanek, Rodrigo Jover, Ernst J Kuipers, Raf Bisschops, and others. 2016. Reporting systems in gastrointestinal endoscopy: Requirements and standards facilitating quality improvement: European Society of Gastrointestinal Endoscopy position statement. *United European gastroenterology journal* 4, 2 (2016), 172–176.

[5] Thomas de Lange, Stig Larsen, and Lars Aabakken. 2005. Image documentation of endoscopic findings in ulcerative colitis: photographs or video clips? *Gastrointestinal Endoscopy* 61, 6 (2005), 715–720.

[6] Facebook. 2018. React. (2018). https://reactjs.org/

[7] FFmpeg. 2018. FFmpeg. (2018). https://www.ffmpeg.org/

[8] Flask. 2018. Flask. (2018). http://flask.pocoo.org/

[9] Steven Hicks. 2018. Demo Docker Repositoryma. (2018). https://hub.docker.com/r/stevenah/mmsys-demo/

[10] Steven Hicks. 2018. Demo Github Repository. (2018). https://github.com/Stevenah/mmsys-demo

[11] Baoyu Jing, Pengtao Xie, and Eric Xing. 2017. On the Automatic Generation of Medical Imaging Reports. *arXiv preprint arXiv:1711.08195* (2017).

[12] Keras. 2018. Keras: The Python Deep Learning library. (2018). https://keras.io/

[13] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[14] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2017. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing* (2017).

[15] NGINX. 2018. NGINX. (2018). https://nginx.org/en/

[16] Nvidia. 2018. Nvidia CUDA Toolkit. (2018). https://developer.nvidia.com/cuda-toolkit

[17] Nvidia. 2018. Nvidia CuDNN. (2018). https://developer.nvidia.com/cudnn

[18] Konstantin Pogorelov, Kristin Ranheim Randel, Carsten Griwodz, Sigrun Losada Eskeland, Thomas de Lange, Dag Johansen, Concetto Spampinato, Duc-Tien Dang-Nguyen, Mathias Lux, Peter Thelin Schmidt, Michael Riegler, and Pål Halvorsen. 2017. KVASIR: A Multi-Class Image Dataset for Computer Aided Gastrointestinal Disease Detection. In *Proc. of MMSYS*. 164–169. https://doi.org/10.1145/3083187.3083212

[19] Konstantin Pogorelov, Michael Riegler, Sigrun Losada Eskeland, Thomas de Lange, Dag Johansen, Carsten Griwodz, Peter Thelin Schmidt, and Pål Halvorsen. 2017. Efficient disease detection in gastrointestinal videos – global features versus neural networks. *Multimedia Tools and Applications* 76, 21 (01 Nov 2017), 22493–22525. https://doi.org/10.1007/s11042-017-4989-y

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

[21] Michael Riegler, Konstantin Pogorelov, Pål Halvorsen, Thomas de Lange, Carsten Griwodz, Peter Thelin Schmidt, Sigrun L. Eskeland, and Dag Johansen. 2016. EIR - Efficient Computer Aided Diagnosis Framework for Gastrointestinal Endoscopies. In *Proc. of CBMI*.

[22] 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. 2016. Computer Aided Disease Detection System for Gastrointestinal Examinations. In *Proc. of MMSys*.

[23] 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. 2016. Computer Aided Disease Detection System for Gastrointestinal Examinations. In *Proc. of MMSYS*. 29:1–29:4. https://doi.org/10.1145/2910017.2910629

[24] Christin Seifert, Aisha Aamir, Aparna Balagopalan, Dhruv Jain, Abhinav Sharma, Sebastian Grottel, and Stefan Gumhold. 2017. Visualizations of Deep Neural Networks in Computer Vision: A Survey. In *Transparent Data Mining for Big and Small Data*. Springer, 123–144.

[25] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. 2016. Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization. *CoRR* abs/1610.02391 (2016). arXiv:1610.02391 http://arxiv.org/abs/1610.02391

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

[27] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. 2014. Striving for Simplicity: The All Convolutional Net. *CoRR* abs/1412.6806 (2014). arXiv:1412.6806 http://arxiv.org/abs/1412.6806

[28] OpenCV team. 2018. Open Source Computer Vision Library (OpenCV). (2018). https://opencv.org/

[29] Rene Vidal, Joan Bruna, Raja Giryes, and Stefano Soatto. 2017. Mathematics of Deep Learning. *arXiv preprint arXiv:1712.04741* (2017).

[30] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proc. of ML*. 2048–2057.

[31] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. 2015. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579* (2015).

[32] Zizhao Zhang, Yuanpu Xie, Fuyong Xing, Mason McGough, and Lin Yang. 2017. Mdnet: A semantically and visually interpretable medical image diagnosis network. In *Proc. of IEEE CVPR*. 6428–6436.

[33] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. 2015. Learning Deep Features for Discriminative Localization. *CoRR* abs/1512.04150 (2015). arXiv:1512.04150 http://arxiv.org/abs/1512.04150

B.3 Paper III — Dissecting Deep Neural Networksfor Better Medical Image Classification and Classification Understanding

Dissecting Deep Neural Networks for Better Medical Image Classification and Classification Understanding

Steven Alexander Hicks[1], Michael Riegler[2,3], Konstantin Pogorelov[1,2], Kim V. Ånonsen[5], Thomas de Lange[5], Dag Johansen[4], Mattis Jeppsson[6], Kristin Ranheim Randel[8], Sigrun Eskeland[7] and Pål Halvorsen[1,2]

[1]*University of Oslo, Norway* [2]*Simula Research Laboratory, Norway* [3]*SimulaMet, Norway*
[4]*UiT - Artic University of Norway* [5]*Oslo University Hospital, Norway*
[6]*ForzaSys AS, Norway* [7]*Bærum Hospital, Norway* [8]*Cancer Registry Norway*

***Abstract*—Neural networks represent a technology that is becoming an important tool with assisting medical doctors in disease detection during patient examinations. At the same time, deep neural networks are also somehow known as being a black box making it hard to understand what is going on inside and how decisions are made. This is especially true if the users are not familiar with the technology and use it out of the box. To make qualified decisions and increase acceptance and trust in the algorithms output, we present a system that allows partially opening the *black box*. This includes an investigation on what the neural network *reacts on*, to both, improve algorithm understanding and data pre-processing resulting in better image classification performance and a stronger intuition of why the system makes a decision. Furthermore, a significant part of a medical expert's time is spent preparing reports after the examinations, and if we already have a system dissecting the network for understanding, the same tool can be used for automatic report generation after the examination. In this paper a system is presented that is able to look into the layers of a deep learning network and present the network's decision in an understandable way to the doctors. Furthermore, we present and discuss how this information can possibly be used for automatic reporting. Our initial results are very promising.**

***Index Terms*—computer aided diagnosis, deep learning**

I. Introduction

Machine learning, in context of image and video analysis using deep learning, is nowadays commonly used in a lot of different fields such as the financial sector, image retrieval or robotics, etc. One important area is to assist medical doctors in disease detection during patient examinations in order to avoid overlooking abnormalities [1]. However, such deep neural networks are also somewhat black boxes (especially for end users like in our case medical doctors) where only a few understand how they make a prediction. To be able to make qualified decisions and to gain the trust of the medical domain, this black box must be opened as the medical doctors often need a rationale of why the system signals a detection. To the best of our knowledge, this is yet an unexplored area of research, especially when it comes to giving explanation to the doctors and involving them in the system pipeline. To improve the black box-understanding, we examine an automatic disease detection system where we dissect a neural network to explore what happens inside its layers and use this information to increase understanding of how it makes a prediction. As a case study, we use live colonoscopy, which is a common gastrointestinal (GI) examination, essential for the diagnosis of most mucosal diseases in the GI tract, particularly diagnosis of colorectal cancer and its precursors. We have previously developed such a live detection system [2]–[4], and compared various machine learning techniques [5]. In the previously developed system, the endoscopist performs the colonoscopy while video frames are automatically analysed. Furthermore, the system provides visual feedback to the doctor if something abnormal is detected [6]. In this paper, we open the black box, with the goal of gaining a deeper understanding and insight into the detection process of a convolutional neural network (CNN) for three different purposes and contributions:

- *better decision support:* The medical doctors often need a reasoning of why the system returns a detection. To the best of our knowledge, this is yet an unexplored area of research, by providing intermediate heat maps from the internal process of the neural network, we gain more insight into how and why a particular prediction is produced.
- *improved data augmentation:* There are several ways to improve and augment input data in order to improve the detection rates. Using the gained intermediate information, we can observe which parts of an image is marked in each layer so that we can identify which regions result in false positives and false negatives. This can be used to improve both classification performance and training data.
- *automatic report generation:* A system dissecting the network for understanding it better, can also be used for automatic examination report generation proposing both images or video clips to include and giving a reason why.

With these target improvements, we present a system looking deeper into a neural network used for GI disease detection. Using the open Kvasir [7] and CVC-968 [8] datasets, we evaluate the base performance and improvements using insights gained by the system. Based on the dissection of

the network, we demonstrate how the system can be used to improve data augmentation by identifying artifacts in the images that confuse the algorithm. This information is used to perform data pre-processing which improves the detection rate and more important the generality of the model. Then, we show how the intermediate network layer information can be used to help medical experts in understanding the decisions made by the network. Finally, we demonstrate how the system can help generating automatic documentation and reports in a standardized way potentially moving more medical expert time from paper work to patient examinations.

II. Related work

Understanding layers of deep learning architectures has been a topic of research for quite some time. Some researchers try to solve this problem by using a more theoretical basis and mathematical approach such as [9], [10]. While this is important, it does not help the end users like medical experts understand the algorithmic decisions and improve their trust in the system. Other researchers try to apply a more visual approach on the problem and visualize layers using different methods such as heat maps or visual representations (texture, heat maps, etc.) [11], [12]. Based on visual content, the next natural step in the process is to generate text from the visual layers to create automatic tags or descriptions of images and videos. In the medical imaging domain, Zhang et al. [13] propose a method to generate automatic reports for automatic image diagnosis networks. The goal is to create semantically meaningful reports. As an example, they used bladder cancer detection. A similar approach can be found in [14]. Both methods can create text from the images and visualizations of the regions the algorithm reacted to.

In comparison to these approaches, our system is not focused on creating automatic text, but rather use the visual attention heat maps to get an understanding of the algorithms decisions, presenting them to medical experts and to improve these decisions by pre-processing the data in a different more effective way. To the best of our knowledge, there is no related work that does this.

Furthermore, pre-processing in deep learning is an often used practice, but it is hard to find a clear description about when to apply which methods. It often depends on the data and the understanding of the data [15], [16]. Therefore, a system that can give visual explanations of the data and show how it connects with the algorithms can be helpful when looking to improve performance.

Finally, our system tries to recommend which images or parts of the videos represent the most important findings. Medical experts indicate that generating automatic text is not the most important feature, but rather to give them a tool that helps them understand the decisions made by the algorithms, and at the same time, supports them in creating reports that represent the case in a unified way [1]. Therefore, taking the user into the loop of system development is an important aspect and requires tools such as the here presented system.

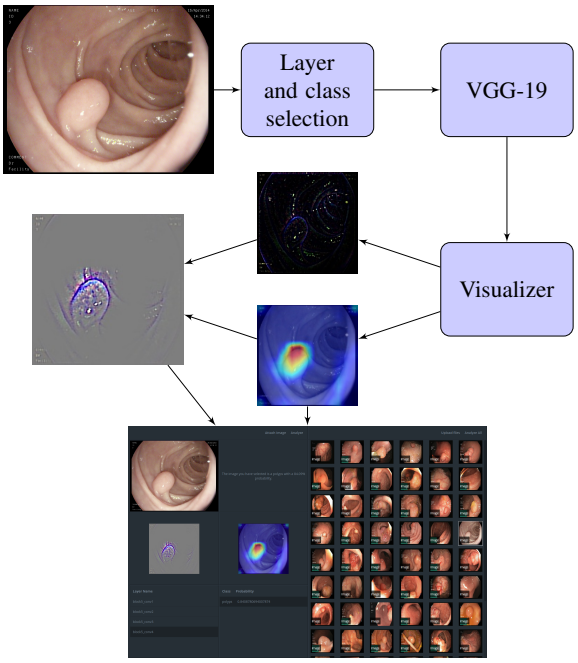
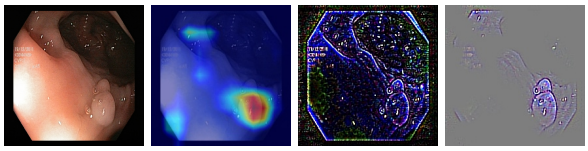


Fig. 1: An overview of how we produce the two visualizations included in the image analysis, and how it is presented in the user interface where a visualization of the different convolutional layers can be selected.

III. System description

The proposed system can be described as a framework with three main functionalities. (i) The system was designed to aid medical doctors in making informed decisions regarding diagnosis of diseases found during examinations, such as diagnosis of disease found in the GI tract during a colonoscopy. (ii) The system creates automatic reports based on the automatic analysis of images and videos and reduces the time spent on the administrative tasks that follow an endoscopic examination, e.g., documentation by written reports. This is shown in Figure 3 where the doctor uses the system to understand the analysis done by the neural network, and use this information to reach a diagnosis and generate the accompanied report. (iii) The system can be used by researchers and engineers designing deep learning architectures such as CNNs to gain a better understanding of the evaluation and reactions of their models, e.g., by understanding which parts of an image confuse the algorithm and if additional pre-processing steps are needed.

The basis of the system is a deep CNN which is used to analyse image or video data to perform different classification tasks, e.g., automatic detection of diseases. This process is made transparent to the users through a tool that dissects the individual layers of a CNN, making it possible to see the basis for the decisions made by the system and on what regions of an image the algorithm activates for a target class. This is a critical piece in building trust among users of the system like medical professionals who need to rely on the systems output



(a) Original image (b) Grad-CAM (c) Saliency Map (d) Guided Grad-CAM

Fig. 2: Image representations used by the reporting system to explain decisions.

without the technical knowledge of the internal working of a CNN. Additionally, it allows for discovering fallacies within the model itself and the dataset used to train it.

Using the guided grad-CAM technique [17], we generate visual representations of an image as it moves through the network, showing what regions of the image correspond to a target class at the point of a selected layer. The process is shown in Figure 1, and starts with the user selecting an input image, target layer and target class using the web-interface (shown at the bottom of Figure 1). Based on the selection, the system generates three visualizations of the image (all shown later in Figure 4). Figure 2a is the original image before any processing. Figure 2b is a grad-CAM (a generalization of class activation map (CAM) [18]) representation of the image which shows what regions of the image correspond the the selected target class. Figure 2c is the saliency map generated using guided back-propagation, this shows the positive activations of the target layer, and is not class specific. Figure 2d depicts the guided grad-CAM representation of the image, which is a combination of the grad-CAM and saliency map. From the three visualizations, the system presents the grad-CAM and the guided grad-CAM to the user.

IV. Implementation Details

The system is accessed through a web-interface, backed up by a RESTful server written in Python (using the micro-framework Flask [19]). As mentioned in section III, the server uses a deep neural network, specifically a CNN using the standard VGG-19 architecture [20], to perform frame analysis. It is important to point out that the architecture used can be changed if needed. The neural network is implemented using the deep learning framework Keras [21] using Tensorflow as a backend [22]. The visualizations are generated on the fly as the user selected an input image, target layer and target class.

V. Neural Network Dissection

As mentioned previously, we use a deep CNN to perform analysis on frames collected from a endoscopic examination taken from a video stream. One of the criteria we set in the previous section was that the system must back up its claims by showing the reasoning behind its decision. To achieve this, the system uses a guided grad-cam [17] approach to visualizing the convolutional layers of a CNN given a target class. Guided grad-cams combine the discriminative properties of CAMs together with a more detailed saliency map [23] to

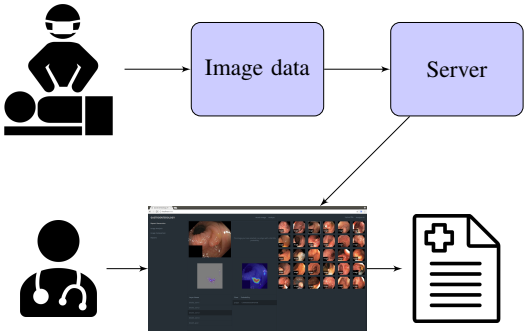


Fig. 3: The expected workflow using the system to analyse data from an endoscopic procedure and produce a written report.

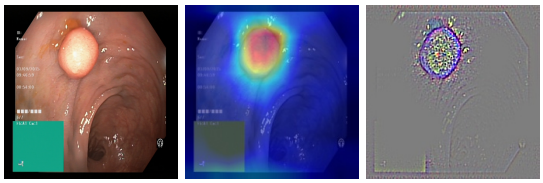
create high quality feature maps, showing detailed localization regions for a target class.

We use the guided grad-cam representation together with a grad-cam to give two perspectives on what the CNN is detecting when making its prediction, which in turn will hopefully distill a higher amount of confidence in the correctness of our network. Principally, the grad-cam and guided grad-cam show the same information, albeit the guided grad-CAM includes a bit more detail, we decided to include both as the grad-CAM is clearer in its explanation. Visualizations are created on a layer by layer basis, making it possible to go back and view the detections made by the lower layers of the network. This might be useful to see what less abstract features are picked up by the network.

The visualization process starts once the user has selected an image, layer and class for further inspection. With an image, target layer and target class selected, we calculate the gradient of the target layer using the loss of the target class in regards to the image. These gradients are globally average pooled to get the weights, which is multiplied with output of the target layer and passed through a relu function to produce the grad-CAM. The grad-CAM is re-sized back to the dimensions of the original image and its values squashed between 0 and 1 before a blue-red heat map filter is applied.

To generate the guided back-propagation saliency map, we start by replacing the activations of our original network with a modified relu function. During back-propagation, a traditional relu would let all gradients whose inputs where larger than 0 pass. We change the relu by adding the additional rule of discarding all gradients that are below 0, thereby only back-propagating the positive influence on the activations. With this modified network, we calculate the gradients of the target layer with respect to the input image, i.e., these gradients represent our saliency map.

Once the grad-CAM and saliency map have been computed, we simply multiply them together to produce the guided grad-CAM representation. This together with the grad-CAM is used in our system.



(a) Original (b) Grad-CAM (c) Grad-CAM

Fig. 4: An image that has been correctly identified as containing a polyp by our CNN, together with the grad-CAM and guided grad-CAM representation.

VI. Understanding the Detections

As mentioned in section IV, we visualize the the convolutional layers of a VGG-19 CNN to understand what each layer detects as the image moves through the network. This is not only useful when trying to detect abnormalities in endoscopic images, but also gives insight into what features the network "thinks" are relevant to a certain class. For example, Figure 4a shows an image containing a polyp located in its upper region. Looking at the grad-CAM representation (Figure 4b), we see the the network correctly identifies the polyp (area in red). Although the used example is quite obvious, this shows that the network, at least, has some knowledge about what a polyp is when it comes to its basic shape. Using the guided grad-CAM representation (Figure 4c), we get a more detailed view of what the network detects, such as texture detection. Looking closely at Figure 4c, we see that the network detects the edge of the polyp, noting that the polyp is raised above the mucosa (blue outline surrounding the polyp).

Figure 4 depicts the image at the last convolutional layer of the network, showing what the network recognizes right before making its prediction. For the most part, this is what we want. But, it may also be useful to look further back in the network to see what less abstract features are detected early in the network. Looking at Figure 5, we see a guided grad-CAM representation of an image at the last convolutional layer of each convolutional block of a VGG-19 CNN. Looking at the first couple of layers (Figures 5b and 5c), we see that the network picks up basic textures of the mucosa. Looking at the latter images, we see that the network starts to see the visual shape of the polyp.

Note that the visualizations are made with respect to a target class, meaning we can see what regions of an image correspond to another class apart form the predicted one. This comes in handy when the network detects multiple classes in a single image. For example, an image may contain signs of ulcerative colitis and polyps, using the visualizations we are able to see the class specific regions of each abnormality. This is also useful when diagnosing issues with the network, understanding why a network "thinks" it detects a certain class that is not there, this will be discussed further in section VII.

VII. Enhancing Input Data for Better Detection

In the previous section, we used the two image representations (grad-CAM and guided grad-CAM) to gain insight into what the network sees when it makes a prediction for a certain class. This not only helps us detect diseases in the GI tract, but can also be used to diagnose issues with a network making incorrect predictions. Using the Kvasir v2 dataset [24], we looked at various samples where the network got confused and mistakenly predicted the wrong class. Figure 6a shows an image of a clean cecum (beginning of the bowel), as part of the *normal cecum* class. The network mistakenly predicted that the image depicts a colon inflicted by ulcerative colitis (inflammatory bowel disease) as part of the *ulcerative colitis* class, with an 86.5% certainty. Using the system to diagnose what the network detects with respect to the class *normal cecum* at the final convolutional layer, we get the grad-CAM (Figure 6b) and guided grad-CAM (Figure 6c) representations, which show us that the algorithm gets confused by the navigation box located in the lower left corner. This indicates that the network has learned the "noise" of an image, and associated it with a class, i.e., it has associated the navigation box with a normal cecum. This is an important observation, as we might be able to use this information to improve the performance of our network.

(a) Kvasir v2

Kvasir v2	PREC	REC	SPEC	ACC	MCC	F1
Non-processed	0.966	0.791	0.736	0.940	0.762	0.758
Navigation box	0.968	0.798	0.753	0.944	0.778	0.778
Navigation box + border	0.968	0.943	0.749	0.943	0.775	0.771

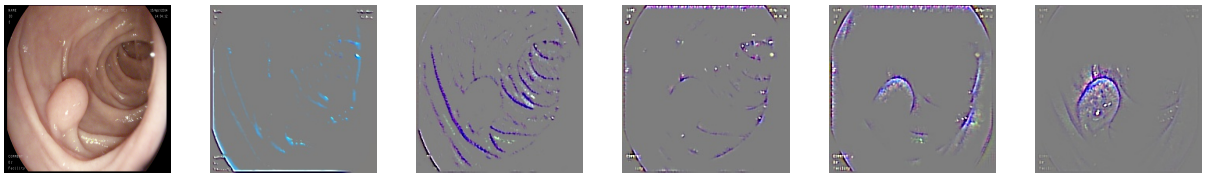
(b) Kvasir v2 + CVC

Kvasir v2 Extra Polyps	PREC	REC	SPEC	ACC	MCC	F1
Non-processed	0.957	0.722	0.673	0.924	0.723	0.702
Navigation box	0.959	0.738	0.691	0.927	0.739	0.719
Navigation box + border	0.964	0.773	0.724	0.937	0.760	0.750

TABLE I: CNN evaluation using 2-fold cross-validation.

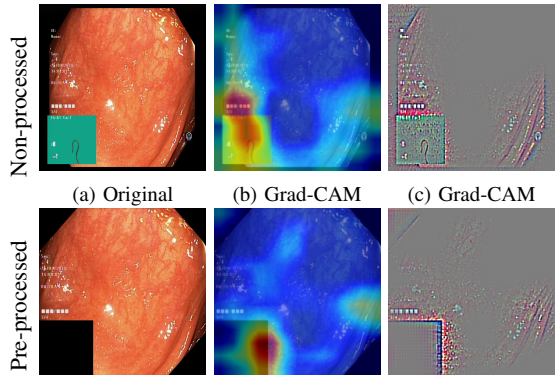
After finding incidents of incorrect predictions because of "noise" in the image, we have two possible options for improving the class detection of our network, i.e., change the network itself or apply additional pre-processing steps to the dataset. For the scope of this paper, we will limit it to applying additional pre-processing steps to the Kvasir version 2 dataset as following: (i) blacked out navigation box and (ii) blacked out navigation box and cropped black borders.

After applying these steps, we re-trained the model and ran the image analysis again. This time we found that that *normal cecum* prediction had fallen down to 28.3%, and the network now correctly classifies it as *ulcerative colitis* with a 53.22% certainty. The change in prediction is promising, but the network still activates on the blacked out navigation box, which causes the still high prediction value fro *ulcerative colitis*, i.e., additional pre-processing steps may lead to better results. In this particular case, a possible reason for the confusion is an imbalance of "noisy" images between the classes, e.g. some classes include many images that have the navigation box located in the lower left corner, while other classes barely contain such images. This is supported by the class *ulcerative colitis*, having few images with a navigation box, often being confused with *polyps* and *normal cecum*, which contain many images with the navigation box.



(a) Original (b) Block 1 (c) Block 2 (d) Block 3 (e) Block 4 (f) Block 5

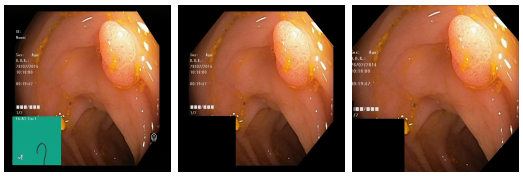
Fig. 5: Guided grad-CAM representation of an image at the last convolutional layer of each convolutional block.



(a) Original (b) Grad-CAM (c) Grad-CAM

(d) Original (e) Grad-CAM (f) Grad-CAM

Fig. 6: An incorrectly identified image with its grad-CAM and guided grad-CAM representation with respect to the class *normal cecum*.



(a) Non processed (b) Navigation box (c) Navigation box + borders

Fig. 7: Examples of data enhancements.

Based on the findings of described in section VII, we trained and evaluated a VGG-19 CNN on three different variations of Kvasir v2; (i) non-processed images (Figure 7a), (ii) navigation box blacked out (Figure 7b), and (iii) navigation box and borders removed (Figure 7c). From this, we observe that the non-processed and pre-processed datasets are distinguishable, but the navigation box/borders removed and the blacked out dataset look quite similar. The difference between the two pre-processed datasets are in the surrounding border. Each dataset was trained and evaluated using 2-fold cross-validation resulting in 500 images used for training and evaluation per class. Table I(a) shows the result of the model evaluation. Looking at the F1 score, we see that the pre-processed datasets perform a couple points better than the non-processed dataset.

As with any neural network, it is important that it generalizes well rather than overfitting on a specific dataset. Therefore, we performed another evaluation on the three dataset variations using additional 400 polyp images taken randomly from the CVC-968 dataset [8] added to the test set. The reason therefore was to show how general the trained model is and that it does not work well on just the dataset used for training (which would be a sign for overfitting). The outcome of this experiment revealed that the non pre-processed dataset was less general than the pre-processed on and most probably dataset specific (overfitted). The result of this evaluation is shown in Table I and further supports the case that the pre-processed datasets perform better than the non-processed. Looking at the individual F1 scores we see that the non-processed dataset fell by 5.6 points, the blacked out dataset fell by 5.9 points, and the border and navigation box removed pre-processing fell by only 2.1 points. This shows that the border and navigation box removed pre-processing training creates a model that generalizes better then the other variations. Even if the general overall performance goes down compared to Table I(b) for a real world scenario a more general model is more important than a dataset specific one.

VIII. Creating Automatic Reports

Being able to understand the output of an algorithm better can also be very important to create automatic reports. The written reporting of medical procedures is an essential, but cumbersome, part of the physicians daily work. Although high quality and completeness of the reports is important [25], it is frequently not complying with existing standards [26]. Within GI endoscopy, both a standardized language and a systematic description of endoscopic findings are needed [27]. Including the findings of a GI endoscopy manually into a simple report takes typically around two minutes [28], [29]. However, complex reports that include description of several abnormalities may take 10-15 minutes or longer to generate. A high-end automatic reporting system based on automatic video analysis has the potential to improve the time needed, the precision and the standardisation of the report in line with the World endoscopy Organisations recommendations [27] and to decrease the time to generate complex reports.

A significant part of a medical expert's time is spent documenting the examination and preparing reports after the examinations, e.g., one study reports that physicians spent 52.9% of the time on direct clinical face time and 37.0% on Electronic Health Record (EHR) and desk work [30], and some claim doctors waste in average 48 minutes per day using electronic medical records [31]. Finally, an automatic reporting system could also help following standards [32]. As mentioned before, a system that can extract information from deep learning layers can also be used for generating automatic

reports by providing images or video clips to include and at the same moment providing a reason why the algorithm came to a certain decision. Nevertheless, this is out of focus for this paper and will be more in focus in future work including studies with medical experts.

IX. Conclusions

Neural networks are widely used in all types of detection, classification and localization of objects in an image or video frame. However, the understanding of how deep neural networks operate and on what their output is based on is in general very limited – even more so among non-technical users. In many domains such as medicine (among others), the users often need to understand why a particular decision is made. To improve the understanding of the internal decision process of deep neural networks and to build trust among its users, we have developed a system that allows to dissect deep neural networks, enabling investigation and understanding of the networks layers and outputs. We presented a detailed explanation about how such a system can be used to increase understanding and performance and evaluated it using two different datasets. The evaluation is showing promising results indicating better performance and generalization of deep learning models after applying improvements based on insights gained using the presented system. Furthermore, we presented and discussed how the intermediate knowledge provided by the system can be used to automatically generate a modifiable report including both text and images increasing the understanding and potential trust of medical experts. For future work we will evaluate the reporting part of the system with the help of medical doctors and improve the automatic report generation part based on this evaluation.

References

[1] M. Riegler, M. Lux, C. Griwodz, C. Spampinato, T. de Lange, S. L. Eskeland, K. Pogorelov, W. Tavanapong, P. T. Schmidt, C. Gurrin, D. Johansen, H. Johansen, and P. Halvorsen, "Multimedia and medicine: Teammates for better disease detection and survival," in *Proc. of ACM MM*, 2016, pp. 968–977.

[2] M. Riegler, K. Pogorelov, J. Markussen, M. Lux, H. K. Stensland, T. de Lange, C. Griwodz, P. Halvorsen, D. Johansen, P. T. Schmidt, and S. L. Eskeland, "Computer aided disease detection system for gastrointestinal examinations," in *Proc. of MMSys*, 2016.

[3] M. Riegler, K. Pogorelov, P. Halvorsen, T. de Lange, C. Griwodz, P. T. Schmidt, S. L. Eskeland, and D. Johansen, "EIR - efficient computer aided diagnosis framework for gastrointestinal endoscopies," in *Proc. of CBMI*, 2016.

[4] K. Pogorelov, M. Riegler, P. Halvorsen, P. T. Schmidt, C. Griwodz, D. Johansen, S. L. Eskeland, and T. de Lange, "GPU-accelerated real-time gastrointestinal diseases detection," in *Proc. of CBMS*, 2016.

[5] K. Pogorelov, M. Riegler, S. L. Eskeland, T. de Lange, D. Johansen, C. Griwodz, P. T. Schmidt, and P. Halvorsen, "Efficient disease detection in gastrointestinal videos – global features versus neural networks," *Multimedia Tools and Applications*, 2017.

[6] M. Riegler, K. Pogorelov, J. Markussen, M. Lux, H. K. Stensland, T. de Lange, C. Griwodz, P. Halvorsen, D. Johansen, P. T. Schmidt, and S. L. Eskeland, "Computer aided disease detection system for gastrointestinal examinations," in *Proc. of MMSYS*, 2016, pp. 29:1–29:4.

[7] K. Pogorelov, K. R. Randel, C. Griwodz, S. L. Eskeland, T. de Lange, D. Johansen, C. Spampinato, D.-T. Dang-Nguyen, M. Lux, P. T. Schmidt, M. Riegler, and P. Halvorsen, "Kvasir: A multi-class image dataset for computer aided gastrointestinal disease detection," in *Proc. of ACM MMSYS*, 2017, pp. 164–169.

[8] J. Bernal and H. Aymeric, "Miccai endoscopic vision challenge polyp detection and segmentation," https://endovissub2017-giana.grand-challenge.org/home/, accessed: 2017-12-11.

[9] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digit. Signal Proc.*, 2017.

[10] R. Vidal, J. Bruna, R. Giryes, and S. Soatto, "Mathematics of deep learning," *arXiv preprint arXiv:1712.04741*, 2017.

[11] C. Seifert, A. Aamir, A. Balagopalan, D. Jain, A. Sharma, S. Grottel, and S. Gumhold, "Visualizations of deep neural networks in computer vision: A survey," in *Transparent Data Mining for Big and Small Data*. Springer, 2017, pp. 123–144.

[12] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *arXiv preprint arXiv:1506.06579*, 2015.

[13] Z. Zhang, Y. Xie, F. Xing, M. McGough, and L. Yang, "Mdnet: A semantically and visually interpretable medical image diagnosis network," in *Proc. of IEEE CVPR*, 2017, pp. 6428–6436.

[14] B. Jing, P. Xie, and E. Xing, "On the automatic generation of medical imaging reports," *arXiv preprint arXiv:1711.08195*, 2017.

[15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[16] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[17] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization," *CoRR*, 2016.

[18] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," *CoRR*, 2015.

[19] "Flask." [Online]. Available: http://flask.pocoo.org/

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[21] "Keras." [Online]. Available: https://keras.io/

[22] "Tensorflow." [Online]. Available: https://www.tensorflow.org/

[23] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, "Striving for simplicity: The all convolutional net," *CoRR*, vol. abs/1412.6806, 2014. [Online]. Available: http://arxiv.org/abs/1412.6806

[24] K. Pogorelov, K. R. Randel, C. Griwodz, S. L. Eskeland, T. de Lange, D. Johansen, C. Spampinato, D.-T. Dang-Nguyen, M. Lux, P. T. Schmidt, M. Riegler, and P. Halvorsen, "Kvasir: A multi-class image dataset for computer aided gastrointestinal disease detection," in *Proc. of MMSYS*, 2017, pp. 164–169.

[25] M. Bretthauer, L. Aabakken, E. Dekker, M. F. Kaminski, T. Rösch, R. Hultcrantz, S. Suchanek, R. Jover, E. J. Kuipers, R. Bisschops *et al.*, "Requirements and standards facilitating quality improvement for reporting systems in gastrointestinal endoscopy: European society of gastrointestinal endoscopy (esge) position statement," *Endoscopy*, vol. 48, no. 3, pp. 291–4, 2016.

[26] T. De Lange, B. Moum, J. Tholfsen, S. Larsen, and L. Aabakken, "Standardization and quality of endoscopy text reports in ulcerative colitis," *Endoscopy*, vol. 35, no. 10, pp. 835–840, 2003.

[27] L. Aabakken, A. N. Barkun, P. B. Cotton, E. Fedorov, M. A. Fujino, E. Ivanova, S.-e. Kudo, K. Kuznetzov, T. Lange, K. Matsuda *et al.*, "Standardized endoscopic reporting," *Journal of gastroenterology and hepatology*, vol. 29, no. 2, pp. 234–240, 2014.

[28] M. Groenen, E. Kuipers, G. van Berge Henegouwen, P. Fockens, and R. Ouwendijk, "Computerisation of endoscopy reports using standard reports and text blocks," *The Netherlands journal of medicine*, 2006.

[29] K. Kuhn, W. Gaus, J. Wechsler, P. Janowitz, J. Tudyka, W. Kratzer, W. Swobodnik, and H. Ditschuneit, "Structured reporting of medical findings: evaluation of a system in gastroenterology," *Methods of information in medicine*, vol. 31, no. 04, pp. 268–274, 1992.

[30] C. Sinsky, L. Colligan, L. Li, M. Prgomet, S. Reynolds, L. Goeders, J. Westbrook, M. Tutty, and G. Blike, "Allocation of physician time in ambulatory practice: A time and motion study in 4 specialties," *Annals of Internal Medicine*, vol. 165, no. 11, pp. 753–760, 2016.

[31] M. CJ, C. FM, W. A, G. RM, M. M, and K. T, "Use of internist,s free time by ambulatory care electronic medical record systems," *JAMA Internal Medicine*, vol. 174, no. 11, pp. 1860–1863, 2014.

[32] T. H. Baron, "Endoscopy: Gastrointestinal endoscopy reporting: time for standardization?" *Nature Reviews Gastroenterology and Hepatology*, vol. 11, no. 3, p. 145, 2014.