UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Clearing the Way in Capsule Endoscopy with Deep Learning and Computer Vision

Doctoral Thesis
Departamento de Comunicaciones

Reinier Noorda

Supervisors:
Valery Naranjo, Vicente Pons Beltrán

Submitted: April, 2022

Acknowledgement

Through the years invested in this doctoral thesis project, I have been able to accomplish the work described in this manuscript and been able to develop myself only with the support, assistance and guidance of individuals who were closely involved during these years.

First and foremost, I want to thank my main supervisor, Valery, for all the effort invested in the supervision of this project and personal guidance in my development as a researcher, for all the teachings, both through her incredible capabilities of simplifying the complicated and by example, and for always looking after my personal well-being. In these years under her supervision, it has amazed me how, where one door closes, she always manages to open a new one, with a vision that extends beyond any borders. It has been a privilege to get to know you personally and to grow under your supervision, Valery, and I hope we can continue to stay in touch.

I would also like to mention my other supervisor, Vicente from hospital La Fe, for all the important medical insights and assuring our work was always in line with medical practice, rather than following an isolated technical path. In this, I would also like to mention Noelia, and especially Andrea, who I closely cooperated with throughout this project for the collection of the data with patient compliance, high quality annotations, and the validation of the methods developed in this work in clinical practice.

Following, I would like to thank all the individuals involved in the WiBEC ITN, for the cooperation and for hosting interesting training schools I had the pleasure to attend. In particular I am grateful to Sofia and Martina, for the exchange of ideas and experiences, Melanie and Salman, for their interest in my work and person especially during my secondment at Ovesco, and Conchi and Teresa, for helpful suggestions and practical support.

I also want to thank my colleagues from the research lab, for the exchange of ideas, knowledge and materials, but also for all the great moments both in and out of the lab, namely Fer, Félix, Andrés, Gabri, Julio, Rocío, Cristian and Elena, as well as Jorge, Adrián B. and Rober. Special thanks go to Sandra, for all her support on the organisation and practical matters surrounding this project, and Adrián, for being a great example for any starting researcher and for all the valuable insights, joy and friendship during all these years of working next to him. I am also grateful to the brilliant students I co-supervised, Pedro and Javier, in their bachelor and master theses respectively, for their efforts, insights and contributions to the conducted research.

Finally, I would like to thank everyone else who contributed through continuous personal support, and by offering me perspective in harder moments. I especially thank my parents, for their unconditional support and teaching me perseverance and goodwill, my siblings, for being there for me and for the valuable lessons learnt developing myself alongside them, Alicia, for her confidence and encouraging me in taking up this challenge, Héctor, for regularly pulling me out of my struggles and misery to completely change my perspective, Lala, for all the company and the energy she provided in otherwise hopeless moments, and Natalia, for keeping faith in me through the most difficult moments and for all her care and support that helped pull me through.

Abstract

Capsule endoscopy (CE) is a widely used, minimally invasive alternative to traditional endoscopy that allows visualisation of the entire small intestine, whereas more invasive procedures cannot easily do this. However, those traditional methods are still commonly the first choice of treatment for gastroenterologists as there are still important challenges surrounding the field of CE. Among others, these include the time consuming video diagnosis following the procedure, the fact that the capsule cannot be actively controlled, lack of consensus on good patient preparation and the high cost. In this doctoral thesis, we aim to extract more information from capsule endoscopy procedures to aid in alleviating these issues from a perspective that appears to be under-represented in current research.

First, and as the main objective in this thesis, we aim to develop an objective, automatic cleanliness evaluation method in CE procedures to aid medical research in patient preparation methods. Namely, even though adequate patient preparation can help to obtain a cleaner intestine and thus better visibility in the resulting videos, studies on the most effective preparation method are conflicting due to the absence of such a method. Therefore, we aim to provide such a method, capable of presenting results on an intuitive scale, with a relatively light-weight novel convolutional neural network architecture at its core. We trained this model on an extensive data set of over 50,000 image patches, collected from 35 different CE procedures, and compared it with state-of-the-art classification methods. From the patch classification results, we developed a method to automatically estimate pixel-level probabilities and deduce cleanliness evaluation scores through automatically learnt thresholds. We then validated our method in a clinical setting on 30 newly collected CE videos, comparing the resulting scores to those independently assigned by human specialists. We obtained the highest classification accuracy for the proposed method (95.23%), with significantly lower average prediction times than for the second-best method. In the validation of our method, we found acceptable agreement with two human specialists compared to interhuman agreement, showing its validity as an objective evaluation method.

Additionally, we aim to automatically detect and localise the tunnel in each frame, in order to help determine the capsule orientation at any given time. For this purpose, we trained an R-CNN based model, namely the light-weight YOLOv3 detector, on a total of 1385 frames, extracted from CE procedures of 10 different patients, achieving a precision of 86.55% combined with a recall of 88.79% on our test set. Extending on this, we additionally aim to visualise

intestinal motility in a manner analogous to a traditional intestinal manometry, solely based on the minimally invasive technique of CE, through aligning the frames with similar orientation and using the bounding box parameters to derive adequate parameters for our tunnel segmentation method. Finally, we calculate the relative tunnel size to construct an equivalent of an intestinal manometry from visual information.

Since we concluded our work, our method for automatic cleanliness evaluation has been used in a still on-going, large-scale study, with in which we actively participate. While much research focuses on automatic detection of pathologies, such as tumors, polyps and bleedings, we hope our work can make a significant contribution to extract more information from CE also in other areas that are often overlooked.

Resumen

La endoscopia capsular (CE) es una ampliamente utilizada alternativa mínimamente invasiva a la endoscopia tradicional, que permite la visualización de todo el intestino delgado, mientras no es posible hacerlo fácilmente con los procedimientos más invasivos. Sin embargo, esos métodos tradicionales aún suelen ser la primera opción de tratamiento, ya que todavía existen desafíos importantes en el campo de la CE, incluyendo el tiempo necesario para el diagnóstico por vídeo después del procedimiento, el hecho de que la cápsula no se puede controlar activamente, la falta de consenso sobre una buena preparación del paciente y el coste alto. En esta tesis doctoral, nuestro objetivo es extraer más información de los procedimientos de endoscopía por cápsula para ayudar a aliviar estos problemas desde una perspectiva que parece estar subrepresentada en la investigación actual.

Primero, como el objetivo principal en esta tesis, pretendemos desarrollar un método de evaluación de la limpieza en procedimientos de CE automático y objetivo para asistir la investigación médica en métodos de preparación de los pacientes. Específicamente, a pesar de que una preparación adecuada del paciente pueda ayudar a obtener una mejor visibilidad, los estudios sobre el método más efectivo son contradictorios debido a la ausencia de tal método. Por lo tanto, pretendemos proporcionar un método de ese tipo, capaz de presentar la limpieza en una escala intuitiva, con una novedosa arquitectura relativamente ligera de una red neuronal convolucional en su núcleo. Entrenamos este modelo en un conjunto de datos extensivo de más de 50,000 parches de imágenes, obtenidos de 35 procedimientos CE diferentes, y lo comparamos con métodos de clasificación del estado del arte. A partir de la clasificación, desarrollamos un método para automáticamente estimar las probabilidades a nivel de píxel y deducir los puntos en la escala de la evaluación de la limpieza a través de umbrales aprendidos. Después, validamos nuestro método en un entorno clínico en 30 videos de CE obtenidos nuevamente, comparando las puntuaciones resultantes con las asignadas de forma independiente por especialistas humanos. Obtuvimos la mayor precisión de clasificación para el método propuesto (95,23%), con tiempos de predicción promedios significativamente más bajos que para el segundo mejor método. En la validación, encontramos un acuerdo aceptable con dos especialistas humanos en comparación con el acuerdo interhumano, mostrando su validez como método de evaluación objetivo.

Adicionalmente, otro objetivo de este trabajo es detectar automáticamente el túnel y ubicar el túnel en cada fotograma. Para este objetivo, entrenamos un

modelo basado en R-CNN, concretamente el detector ligero YOLOv3, en un total de 1385 fotogramas, extraídos de procedimientos de CE de 10 pacientes diferentes. De tal manera, alcanzamos una precisión del 86,55% y una recuperación del 88,79% en nuestro conjunto de datos de test. Ampliando este objetivo, también pretendemos visualizar la motilidad intestinal de una manera análoga a una manometría intestinal tradicional, basada únicamente en la técnica mínimamente invasiva de CE. Para esto, alineamos los fotogramas con similar orientación y derivamos los parámetros adecuados para nuestro método de segmentación de las propiedades del rectángulo delimitador del túnel. Finalmente, calculamos el tamaño relativo del túnel para construir un equivalente de una manometría intestinal a partir de información visual.

Desde que concluimos nuestro trabajo, nuestro método para la evaluación automática de la limpieza se ha utilizado en un estudio a gran escala aún en curso, en el que participamos activamente. Mientras gran parte de la investigación se centra en la detección automática de patologías, como tumores, pólipos y hemorragias, esperamos que nuestro trabajo pueda hacer una contribución significativa para extraer más información de la CE también en otras áreas frecuentemente subestimadas.

Resum

L'endoscòpia capsular (CE) és una àmpliament utilitzada alternativa mínimament invasiva a l'endoscòpia tradicional, que permet la visualització de tot l'intestí prim, mentre no és possible fer-lo fàcilment amb els procediments més invasius. No obstant això, aqueixos mètodes tradicionals encara solen ser la primera opció de tractament, ja que encara existeixen desafiaments importants en el camp de la CE, incloent el temps necessari per al diagnòstic per vídeo després del procediment, el fet que la càpsula no es pot controlar activament, la falta de consens sobre una bona preparació del pacient i el cost alt. En aquesta tesi doctoral, el nostre objectiu és extraure més informació dels procediments de endoscopía per càpsula per a ajudar a alleujar aquests problemes des d'una perspectiva que sembla estar subrepresentada en la investigació actual.

Primer, com l'objectiu principal en aquesta tesi, pretenem desenvolupar un mètode d'avaluació de la neteja en procediments de CE automàtic i objectiu per a assistir la investigació mèdica en mètodes de preparació dels pacients. Específicament, a pesar que una preparació adequada del pacient puga ajudar a obtindre una millor visibilitat, els estudis sobre el mètode més efectiu són contradictoris a causa de l'absència de tal mètode. Per tant, pretenem proporcionar un mètode d'aqueix tipus, capaç de presentar la neteja en una escala intuïtiva, amb una nova arquitectura relativament lleugera d'una xarxa neuronal convolucional en el seu nucli. Entrenem aquest model en un conjunt de dades extensiu de més de 50,000 pegats d'imatges, obtinguts de 35 procediments CE diferents, i el comparem amb mètodes de classificació de l'estat de l'art. A partir de la classificació, desenvolupem un mètode per a automàticament estimar les probabilitats a nivell de píxel i deduir els punts en l'escala de l'avaluació de la neteja a través de llindars apresos. Després, validem el nostre mètode en un entorn clínic en 30 vídeos de CE obtinguts novament, comparant les puntuacions resultants amb les assignades de manera independent per especialistes humans. Vam obtindre la major precisió de classificació per al mètode proposat (95,23%), amb temps de predicció mitjanes significativament més baixos que per al segon millor mètode. En la validació, trobem un acord acceptable amb dos especialistes humans en comparació amb l'acord interhumà, mostrant la seua validesa com a mètode d'avaluació objectiu.

Addicionalment, un altre objectiu d'aquest treball és detectar automàticament el túnel i situar el túnel en cada fotograma. Per a aquest objectiu, entrenem un model basat en R-CNN, concretament el detector lleuger YOLOv3, en un total de 1385 fotogrames, extrets de procediments de CE de 10 pacients difer-

ents. De tal manera, aconseguim una precisió del 86,55% i una recuperació del 88,79% en el nostre conjunt de dades de test. Ampliant aquest objectiu, també pretenem visualitzar la motilitat intestinal d'una manera anàloga a una manometría intestinal tradicional, basada únicament en la tècnica mínimament invasiva de CE. Per a això, alineem els fotogrames amb similar orientació i derivem els paràmetres adequats per al nostre mètode de segmentació de les propietats del rectangle delimitador del túnel. Finalment, calculem la grandària relativa del túnel per a construir un equivalent d'una manometría intestinal a partir d'informació visual.

Des que concloem el nostre treball, el nostre mètode per a l'avaluació automàtica de la neteja s'ha utilitzat en un estudi a gran escala encara en curs, en el qual participem activament. Mentre gran part de la investigació se centra en la detecció automàtica de patologies, com a tumors, pòlips i hemorràgies, esperem que el nostre treball puga fer una contribució significativa per a extraure més informació de la CE també en altres àrees sovint subestimades.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 2
- 1.2 Objectives . . . 5
- 1.3 Outline . . . 6

2 Capsule Endoscopy **7**
- 2.1 The Procedure . . . 8
- 2.2 PillCam SB 3 . . . 9
  - 2.2.1 Hardware Component . . . 9
  - 2.2.2 Video Recordings . . . 10
- 2.3 Data Collection . . . 11
  - 2.3.1 Extraction Procedure . . . 11
  - 2.3.2 Preprocessing . . . 12
- 2.4 The Future . . . 12
  - 2.4.1 Active Capsules . . . 13
  - 2.4.2 Computer-Aided Diagnosis . . . 14

3 Assessment of Visibility in Capsule Endoscopy Videos **17**
- 3.1 Background and Motivation . . . 19
- 3.2 Overall Intestinal Content Assessment Method . . . 21
  - 3.2.1 Intestinal Content Detection . . . 21
  - 3.2.2 Data Partitioning . . . 25
  - 3.2.3 Model Evaluation . . . 28
  - 3.2.4 Visualisation of Results . . . 29
  - 3.2.5 Quantifying Detection Results . . . 30
  - 3.2.6 Validation in Clinical Setting . . . 31
- 3.3 Approach Based on Hand-crafted Features and Classification . . . 33
  - 3.3.1 Green-Red Colour Features . . . 34
  - 3.3.2 Local Binary Patterns . . . 36
  - 3.3.3 Support Vector Machine Classification . . . 39
- 3.4 Approach Based on Convolutional Neural Networks (CNNs) . . . 42
  - 3.4.1 Transfer Learning . . . 43
  - 3.4.2 Novel CNN Architecture for Intestinal Content Detection 48
- 3.5 Experiments and Results . . . 49
  - 3.5.1 Data Preparation . . . 49

3.5.2 Hand-crafted Features Compared to Fine-tuning VGG architectures . . . 53
3.5.3 Novel CNN architecture . . . 55
3.5.4 Validation . . . 64
3.6 Discussion . . . 65

4 Determination of Capsule Orientation for Motility Analysis 69
4.1 Background and Motivation . . . 70
4.2 Tunnel Detection . . . 72
4.3 R-CNN Fundamentals . . . 74
4.3.1 Introduction . . . 74
4.3.2 Informed Region of Interest Proposals . . . 79
4.3.3 RoI Pooling . . . 81
4.3.4 Bounding Box Anchors . . . 84
4.3.5 Bounding Box Regression . . . 86
4.3.6 Loss Functions . . . 88
4.3.7 Model Evaluation . . . 89
4.4 Proposal-based Detectors . . . 90
4.5 Regression-based Detectors . . . 92
4.6 You Only Look Once (v3) . . . 95
4.7 Experiments and Results . . . 96
4.7.1 Data Preparation . . . 96
4.7.2 Tunnel Detection . . . 99
4.7.3 Optical Manometry Approximation . . . 101
4.8 Discussion . . . 108

5 Conclusion 117

Publications 121

Bibliography 123

A Convolutional Neural Networks 133
A.1 Learning Procedure . . . 134
A.2 Architecture . . . 136
A.3 Convolution . . . 140
A.4 Layer Types . . . 145
A.4.1 Convolutional . . . 146
A.4.2 Pooling . . . 150
A.4.3 Drop-out . . . 152
A.4.4 Batch Normalisation . . . 152

A.4.5 Softmax 153
A.5 Loss Functions 153
A.5.1 Cross-Entropy 154
A.5.2 Kullback-Leibler Divergence 155
A.5.3 Hinge Loss 156
A.5.4 L_1-loss and L_2-loss 156
A.6 Activation Functions 158
A.6.1 Logistic function 158
A.6.2 Tan-h 159
A.6.3 ReLU 159
A.6.4 Leaky ReLU 159
A.6.5 ELU 160
A.6.6 Maxout 160
A.7 Weights Initialisation 160
A.8 Back-propagation 162
A.9 Gradient Descent 167
A.9.1 Momentum 170
A.9.2 Adam 170
A.9.3 Nadam 171

Chapter 1

Introduction

In this chapter we discuss the motivations behind the work in this thesis and identify the objectives, while we also provide the general outline and explain why this differs somewhat from standard convenience.

Contents

1.1 Motivation **2**
1.2 Objectives **5**
1.3 Outline **6**

1.1 Motivation

Capsule endoscopy is an endoscopic method of diagnosis that first emerged in the early years of this century as a minimally invasive method of diagnosis of the small intestine. Throughout its two decennia of existence, the method was further developed and advanced into hospitals world-wide. While the term can be used for methods of visualisation of any part of the gastro-intestinal system in general through a capsule-shaped device, generally consisting of at least a printed circuit board (PCB) and a camera, in this work we use it to refer exclusively to the method aimed at visualisation of the small intestine. Due to the size of the capsule and other practical reasons, storage of the recorded video usually happens on an external device to which the recorded data is transmitted wirelessly. Therefore this method is also referred to as wireless capsule endoscpoy (WCE), which can be often used interchangeably. In this work we will the shorter, more generic term capsule endoscopy, simply abbreviated CE.

One of the greatest advantages of capsule endoscopy compared to earlier endoscopy techniques, hereafter called traditional techniques, is that the procedure is minimally invasive for the patient. While the pill needs to be activated by a doctor and the patient needs to be observed during the initial phase, the patient does not need sedation and can carry out the rest of his daily activities as normally while the capsule makes its way through the intestine. As the data is recorded on an external device, the patience does not need to retrieve the swallowed device afterwards and can simply return the external device with the recorded video.

Not only is CE minimally invasive compared to other endoscopy techniques, but another important advantage was also the first method that allowed visualisation of the entire small intestine, in contrast to traditional endoscopy techniques that only allowed for the visualisation of the stomach and the first part of the small intestine (upper gastro-intestinal endoscopy) or the colon and the last part of the small intestine (colonoscopy). Just shortly after capsule endoscopy was introduced, another new method of enteroscopy was introduced that allowed for visualisation of the entire intestinal tract, with real-time control and possibility of intervention, just like traditional endoscopy techniques [99]. While this method may be a better option in the knowledge that a patient may require surgery, it has the same level of invasiveness as the traditional endoscopy methods. Capsule endoscopy is therefore still the least invasive method for the purpose of diagnosis.

Despite the obvious advantages, there are also important disadvantages. One of these, is that the relevant parts of the videos resulting from CE procedures are on average between 3 and 4 hours long and to analyse this material manually, a trained gastroenterologist needs to spend approximately 1.5 hour per video diagnosis. Not only is this costly both in time and money, it has also been shown to lead to fatigue due to which pathologies can be missed. Therefore, an important area of research on capsule endoscopy focuses on the automatic detection of pathologies to aid the diagnosis in a direct fashion. Even though we also started working on this initially, focusing on detection of intestinal bleedings, we soon realised that this area of research is quite saturated, while in cooperation with hospital La Fe in Valencia we identified other important areas of research that could contribute to the advancement of the technology in to a similar extent and are in fact under-represented in contemporary research.

Namely, in contrast to traditional endoscopy techniques, capsules move through the intestine passively as opposed to being actively controlled or steered. Due to this, there is no option to focus on a certain area or to purposely revisit an area seen before. While there is already little sense in disposing of an irrigation system to clear up liquids or objects that compromise visibility, due to a lack of control over the capsule throughout the procedure, by design they do not have the capacity for this either. As a consequence, some of the videos cannot be analysed properly due to the presence of intestinal content, such as bile, bubbles and remainders of food, which thus prevents a clear vision of the mucosa in those sections of the video. To overcome this disadvantage and work towards making capsule endoscopy a more widely applicable diagnostic method, this is the main problem we focus on in this thesis project.

To optimise the chances for proper analysis of the CE procedure and to prevent the procedure from being rendered useless, different clinical centres around the world employ different methods of patient preparation. Current popular methods of patient preparation are a clear liquid diet and osmotic laxatives, such as polymer laxatives (polyethylene glycol-electrolyte solutions) and saline laxatives (aqueous sodium phosphates solutions) [64]. All of these solutions work through drawing large amounts of liquid to the intestine, thus softening the stool and causing watery bowel movements, clearing the stool from the intestine. In the case of the clear liquid diet, this is done through ingesting large amounts of water (up to 4 liters) orally. The laxatives work by drawing water into the intestines and retaining it through osmosis, which can cause dehydration as a side effect. The saline laxatives essentially consists of salts

in liquids and thus acts very rapidly, within 30 minutes to 3 hours. The type of polymer laxatives instead consists of large molecules that cause the stool to hold and retain water [78]. This type usually has a slower effect and is therefore better tolerated, with results expected in about 6 hours.

As these methods of patient preparation correspond to different levels of tolerance for the patient, ideally the methods with lower tolerance should only be used if they have an obvious advantage to the methods that are better tolerated. Although the manufacturer of the most widely used capsule endoscope only states to put the patient on a liquid diet in its official manual, it is often argued that best results are obtained in combination with laxatives. Several studies have attempted to compare these, but there is currently no consensus due to the absence of an accurate and objective evaluation method. The most often employed evaluation method in this context is human evaluation. Such evaluation is performed by having several experts rate the cleanliness of the CE videos, such as in the work by Lai et al. [44]. These methods suffer from high subjectivity that is inherent to the human mind, as human judgement is influenced by other factors such as mood and fatigue. This could result in a single video being rated differently not only between two different persons (inter-rater), but also by the same person when asked to rate the same video twice at different moments in time (intra-rater). To solve the issue of subjectivity, a computerised method was proposed by Klein et al. [41]. Computerised methods that always judge videos the same way by exactly the same criteria are guaranteed to always obtain the same results for the same video and therefore have a high intra-rater reliability. The problem with computerised methods is accuracy, as they may not evaluate images the same way as a human expert would. That method, for example, only considered global frame information, such as the dominant colour values per frame [41], without considering textural differences information and locally occurring colours in each frame, while exactly the combination appears to be of influence in problem.

Other than the above, we identified issues with capsule localisation throughout the CE procedure. Namely, due to a lack of active control, it often remains unclear where the capsule is located at a random point in time of the CE procedure. Although localisation methods using RF-sensors that are attached to the patient's body exist, they are often found to perform poorly in practice. In this context, determining the orientation of the capsule can be a significant help in methods that attempt to determine this from visual information, which, in turn, has shown to be of added value to RF-based localisation methods.

One of the other developments that give hope in this matter is research after actively controlled capsules. This partially inspired our secondary objective,

as such capsules would require a method of detecting the way forward, which we later introduce as detection of the *tunnel*. At the same time, the issues we encountered with currently investigated active control capsules directed our attention to research on intestinal motility, as one of the main obstacles in that field, while it is the driving force for the uncontrolled capsules. Not only does this have a significant impact on the locomotion of the capsule, but it is also related to many intestinal diseases. The diagnosis of motility is rarely performed through capsule endoscopy, however, although we believe much more information can be extracted from this procedure, which can be helpful both in diagnosis and for technical purposes related to the capsule.

1.2 Objectives

Based on the prior discussion, we defined two objectives we aim to achieve in this work. Namely, our main objective is to develop a tool that can evaluate the amount of intestinal content automatically, objectively and accurately. To achieve this objective, most importantly, we aim to design a highly accurate intestinal content detection method through several iterations of experiments and research. We then aim to designed a method that automatically converts the detection results to per-frame and per-video cleanliness scores, designed to be equivalent to the medical scales used in prior medical studies. Finally, we conclude this objective by performing an extensive validation in a clinical setting to assure acceptable performance of our method in clinical use.

Additionally, we defined a secondary objective for our work, namely to automatically detect and localise the tunnel, both in state of contraction and in state of relaxation of the intestinal muscles, in order to help determine the capsule orientation at any given time. This can prove useful in the localisation of the capsule, navigation algorithms for future capsules and alignment of frames. Following up on this, we aim to benefit from the latter purpose of this method, the alignment of frames, to visualise intestinal motility directly from the CE recording. Although currently the main benefit of the motility visualisation method will be to aid in diagnosis of motility disorders using solely minimally invasive techniques, we believe it can also be useful in future methods to improve mechanisms in the active locomotion in future capsules, where intestinal motility is one of the main obstacles encountered.

1.3 Outline

While our objectives share a related general purpose, they are substantially different from each other at the same time. Therefore, throughout the research work carried out in this thesis, for each of the objectives we based our work on separate research on important state of the art methods and techniques, and carried out separate which led to separate discussions. Therefore, we decided to structure this thesis differently from standard convenience.

We first introduce the domain of capsule endoscopy, in which all of our work aims to make progress, in Chapter 2. We then dedicate the next two chapters to our main objectives, namely the assessment of the degree of visibility in Chapter 3 and the determination of capsule orientation for intestinal motility visualisation in Chapter 4. Within each of these chapters, we discuss the specific problems, prior research in the field, the theory of our overall methods, describe the experiments carried out in each context and discuss the results. Broader background on the theory that is shared among the methods in both chapters, but which some readers of the target audience may already be familiar with, is included as an appendix in Appendix A. Finally, we end with a general conclusion on all of the work carried out in the context of this thesis in Chapter 5, followed by the publications resulting from our work.

Chapter 2

Capsule Endoscopy

The work in this thesis project is based on the diagnostic procedure of capsule endoscopy. Therefore, as a part of the motivation of this thesis, we already briefly described the procedure and discussed its advantages and disadvantages compared to more traditional methods in the previous chapter. In this chapter, we discuss the general procedure, the characteristics of the specific model of the capsule endoscope that we used in all of our work, the PillCam SB 3, and describe the manner in which we extracted our data for use in our methods. We also briefly discuss the future of capsule endoscopy and the relevance of our work in light of this.

Contents

2.1 The Procedure **8**
2.2 PillCam SB 3 . . . . . **9**
2.2.1 Hardware Component 9
2.2.2 Video Recordings 10
2.3 Data Collection **11**
2.3.1 Extraction Procedure 11
2.3.2 Preprocessing 12
2.4 The Future **12**
2.4.1 Active Capsules 13
2.4.2 Computer-Aided Diagnosis 14

2.1 The Procedure

The procedure of capsule endoscopy in fact starts before the patient swallows the pill-sized device, namely when the patient's bowel is prepared. If the patient were allowed to ingest food as normally, the view of the mucosa would be obstructed by bile, intestinal liquid and remainders of food, preventing a proper diagnosis. Therefore, in first instance, the patient will have to stop eating or ingesting liquids at least 12 hours before ingesting the capsule endoscope [14].

However, as we will discuss in detail in the next chapter, a further preparation is recommended. This preparation may consist of clear liquid diets and/or specific laxatives. Studies have been performed on the effectiveness of these preparation methods, but have thus far not been conclusive. This one of the major issues in capsule endoscopy that we address in this work.

For the start of the actual procedure, the patient will be required to come to the hospital to ingest the pill-shaped device, as shown in Figure 2.1, with instructions from and under supervision of a gastroenterologist. There, the patient is instructed about the procedure by gastroenterologists. Although differences between models and manufacturers may exist, the device generally transmits video to an external receiver that is worn around the waist with a belt. Some models include sensors to determine the location of the pill that corresponds to the recorded video, but this has been shown not to be satisfactory.

As soon as the device is activated and starts recording, the patient swallows the device with water, while he preferably remains in the room until the pill has successfully made its way to the small intestine. From that moment, the patient can go home or to work and continue his daily life as normally. The device commonly takes anywhere between 6 and 10 hours for to complete its trajectory through the small intestine, after which the patient will be required to return the external storage device for reading of the diagnosis by gastroenterologists. Depending on the diagnosis, the patient may be required to return for surgery of the affected part of the intestine.

Figure 2.1: The PillCam SB 3, which is the capsule endoscope model used in this work.

2.2 PillCam SB 3

2.2.1 Hardware Component

According to the manufacturer, the PillCam SB 3 system consists of the specific SB 3 capsule, the SB 3 sensor belt, the SB 3 recorder and the SB 3 sensor array. The sensor belt has been designed to store the recorder comfortably and wear it comfortably over a single layer of clothing around the waist while the patient continues his daily life. The recorder runs on proprietary firmware and is mainly required for storing the video for its later interpretation by a gastroenterologist. However, it is also equipped with small display that can show the status of the capsule, as well as the live recording for usage by a gastroenterologist, mainly to ensure that the device passes through the stomach successfully. The sensor array can be attached to the belt and features 8 leads that are placed carefully on the patient's body in order and receive transmission data, which is then combined to estimate the location of the capsule in two dimensions. Each of sensor leads has a diameter of 40 mm.

In this work, we focus on the capsule, and mainly on its camera. The PillCam SB 3 capsule is the third capsule endoscope model from its manufacturer, Given Imaging, although this manufacturer was acquired by Covidien in 2014, while Covidien was acquired by Medtronic in 2015 [16]. Figure 2.1 shows a photograph of this model. The following information is as provided by the manufacturer [59]. It is cilinder-shaped with a length of 26.2 mm, a diameter of 11.4 mm and a weight of 3.0 mm. The encapsulation of the device is made out

of biocompatible plastic, while it also has mercury-free silver-oxide batteries and the system as a whole can operate at temperatures between 20 and 40 degrees, as necessary in the small intestine.. To provide the necessary light for recording in the dark small intestine, it has 4 white light emitting diodes on each side. The reported viewing angle as measured by the ISO-18600-3 standard is 156 degrees. Most important for the work in this thesis is the frame rate of the device, which is variable between 2 and 6 frames per second, depending on how fast the capsule is moving at each moment. The operating time is at least 8 hours.

2.2.2 Video Recordings

Videos from the capsule endoscope are recorded in a proprietary file format. This file format is read by the accompanying RAPID Reader software, which provides the necessary tools to playback the videos, skip through them. To facilitate reading the videos for gastroenterologists, it also provides visual aids and basic detectors, such as a colour bar, an integrated bleeding detector and a detector of the different segments of the intestine. Most importantly in our work, the software allows saving fragments or single frames, annotated or raw, in MPEG-format or JPEG-format respectively, which allows to process them further as these are open formats that are easily read through readily available functions in MATLAB and common software libraries.

In this work, we exclusively used the exported videos from RAPID Reader. The MPEG-format in which these videos are saved, contain a header which defines general parameters for the video such as frame rate and encoding scheme. This encoding scheme then encodes the pixel colour values in an efficient manner to save storage space, which can be decoded to obtain a single tuple of colour values per pixel. These values define a colour according to a specific colour scheme or *colour space*, which depends on the encoding scheme. In the case of the encoding schemes used in MPEG, colour values are encoded in the RGB colour space, which encodes colours in a 3-dimensional tuple of red, green and blue intensity values. As colour spaces play an important feature extraction and were therefore also an important part of our research, we discuss them in more detail in Section 3.3.1.

2.3 Data Collection

2.3.1 Extraction Procedure

As briefly discussed, we extracted all data used in this work from videos that we exported through RAPID Reader. Namely, early in our experiments we noticed that the frames exported separately as JPEG-files through RAPID Reader had a higher resolution (or lower compression rate) than the MPEG-videos. To make our models perform well in the analysis of those videos and make them more robust in general, we thus decided to exclusively train our models on the frames extracted from lower-quality videos. These were extracted by an experienced gastroenterologist. In the cases where we required the entire small intestine visualisation, the gastroenterologist was instructed to mark the entrance points to and exit points from the small intestine in the RAPID Reader software, after which the video fragment in between those locations could be exported and saved in MPEG-format through a feature of the software that exports the fragment in between marked frames. These are the videos we then considered as the entire original videos in our experiments. In cases where we needed specific data for training, such as frames showing pathologies, the gastroenterologist was instructed to select those frames and export the fragment around it through another feature that saves the fragment to an MPEG-video with equal format and compression settings as the other feature.

While the capsule endoscopy videos are recorded with a variable frame rate of 2 to 6 frames per second (FPS), the exported videos in fact appear to show each frame for an equal amount of time. Namely, each frame of the exported output video seemed to appear 5 times in the output video, while the frame rate reported in the header is 25 FPS. While it seems this would correspond to a rate of 5 unique frames per second, during our extraction procedure we also noticed that not all frames from the original video are stored in the output file. Although we cannot be sure about the number of frames that do not actually appear in the output file, we consider it could be a substantial number, as inputs recorded at 2 to 6 FPS of a 6 hours trajectory were often reduced to videos of 15 minutes. We do have to note here that also in RAPID Reader these videos actually played back with an increased frame rate with respect to the original recording.

2.3.2 Preprocessing

The MPEG-videos exported through RAPID Reader required further processing for us to use them in our models. To start with, as all of our models analyse videos frame by frame, we wrote a script that automatically extracts still video frames from the MPEG-videos at predefined intervals, which are then saved into JPEG-videos with the minimum compression rate to avoid further data loss. These frames then served as input to our annotation tools to extract relevant data to use those in training or testing of our models.

In this work we used three different types of annotations: image-level annotations, annotation of regular regions of predefined size (patches) and bounding box annotation with dimensions as chosen by the annotator. In all cases, annotations were performed by at least one experienced gastroenterologist. Our annotation tools then saved the annotations in matching file names, in order to extract use them as labels or further extract image regions where required for our models.

Only in the case of annotation of patches we would further process the data. Namely, in this case we would extract all of the annotated regions from the original images and save them with minimal further compression into separate files, along with their annotations by saving them into folders that matched a specific annotation. The regions that were not annotated, which as instructed were usually patches that could not be exclusively assigned to one of the classes, were discarded as uninformative and therefore not further used in the procedure.

In all cases, we further ensured that data from a single patient was exclusively contained in the union of the training and validation set or in the test set, and thus never in both at the same time. We did this to assure in the evaluation of our models that they would generalise well not only to new data from the same patients, but also to data from patients whose intestinal tract had not been observed by our model in the training phase.

2.4 The Future

The future branches of CE mainly focus on two interesting topics: the development of active capsules, which can be controlled and steered through the intestine, and automatic offline or real-time diagnosis. In some ideas, the real-time diagnosis is integrated in a new capsule endoscope or medical device, such

as one that detects gastro-intestinal bleeding [76]. Nonetheless, we will discuss each of these ideas separately.

2.4.1 Active Capsules

The development of active capsules is an idea that has risen from the desire to have the same possibilities as in manual endoscopy procedures, but with the same level of minimal invasiveness for the patient as current methods of capsule endoscopy. While this thesis touches this branch of development as well in the consideration of the methods to develop, it does not have its main context in this branch as we do not focus on the hardware development. To give the reader a general idea of this branch of development, however, and the plausibility of possibilities of software methods on-board of active capsule in the future, we reviewed the work that has currently been done on the implementation of the steering mechanisms.

Essentially, in literature we have found three options of driving forces for active navigation: magnetism, biology-inspired approaches and electro-impulses. Among these options, magnetism appears the most plausible and is therefore the most widely investigated technique. The idea behind this technique is to have a small magnet built into the capsule, which is moved by the attracting forces of a magnet outside the body. While this magnet could be directly controlled manually, it is often built into a robotic system which is controlled by a through a remote control system that allows for a more intuitive control by a doctor, in combination with a provided view from the camera of the capsule. This idea was first investigated in combination with an existing external robotic system for cardiovascular procedures [9], while another study focused on developing a navigation method with improved usability when steering a capsule through a liquid-filled stomach [38]. Another study also compared navigation methods [13], showing that steering the robot arm through a robotic system was more accurate than manually moving the magnet

Other options have been suggested that implement active locomotion on the side of the capsule instead, requiring an increased power supply in the already scarce space. For this, biology-inspired approaches have been suggested that rely on earthworm-like movements [7] or mechanical legs [66]. Electro-impulses have also been investigated in earlier work [61], where it was proposed to stimulate contraction of the intestinal wall through electro-impulses given by electrodes attached to the capsule, working in symbiosis with the natural

peristalsis of the bowel. The advantage of all these approaches is that no external equipment is required. A detailed review of all these and other methods for active capsules has been performed by Keller et al. [39].

2.4.2 Computer-Aided Diagnosis

The second branch of future CE development focuses mainly on machine learning methods to perform automatic detection of pathologies and other events of interest, they are not intended to replace human diagnosis as is sometimes suggested, but instead as a method of improving efficiency and medical yield of diagnosis. These methods have come a long way and are slowly being integrated in medical practice as computer-aided diagnosis (CAD) tools. This branch of development the one in which this thesis has its main context.

While the oldest methods focus on simpler manually determined rules that are applied programmatically, current methods are define more abstract mathematical expressions and rules that are applied to input in a generic manner in order to extract desired features from the input. These features are then fed to a method of regression that, when correctly trained, can learn the parameters to assign scores for the input belonging to a certain class and thus effectively learns how to classify samples. Through methods of sampling, the combination of these can be used for object detection in images. These methods are *machine learning* methods, as the algorithm learns the parameters from the input as opposed to a human defining the parameters based on his or her own observations. In this work we refer to these methods as traditional machine learning methods, as opposed to the later branch of methods that we discuss in the next paragraph. We used these methods in our work and therefore discuss them in further detail in Section 3.3.

The later branch of machine learning methods abstracts this idea even further by not only learning the parameters for classification, but also learning the method of feature extraction altogether by defining a generic family of functions for which it learns the parameters. These are also called end-to-end learning approaches. In image processing, within these methods the convolutional neural networks (CNNs) have gained significant ground. We also used and therefore discuss them in detail in Appendix A. Within this branch, the latest methods also incorporate the sampling method in order to turn them into complete detectors themselves. Although this seems more convenient, we only used these in parts of our work as we will explain in Chapter 3, and will therefore discuss them later on in Chapter 4.

In all of the relevant chapters referred to above, we will also discuss the methods that have been used for automatic detection of the relevant intestinal events in prior work.

Chapter 3

Assessment of Visibility in Capsule Endoscopy Videos

In this chapter, we describe the methods we use for the main objective of our thesis to develop a tool that can evaluate the amount of intestinal content automatically, objectively and accurately. We describe the different approaches we investigated and compared for the intestinal content detection in the first step, building up to the approach we eventually developed and integrated in our overall method, and describe our approach to validation in a clinical setting. Finally, we describe the individual experiments in detail and discuss the results.

This chapter contains excerpts from the following works that were previously published in the context of this thesis:

- **Noorda, R.**; Nevárez, A.; Colomer, A.; Pons Beltrán, V.; Naranjo Ornedo, V. Automatic evaluation of degree of cleanliness in capsule endoscopy based on a novel CNN architecture. *Scientific Reports* **10**, 17706 (2020).
- **Noorda, R.**; Nevárez, A.; Colomer, A.; Naranjo, V.; Pons Beltrán, V. (2020). Automatic Detection of Intestinal Content to Evaluate Visibility in Capsule Endoscopy. In *IEEE 13th International Symposium on Medical Information Communication Technology.*

Contents

3.1 Background and Motivation 19
3.2 Overall Intestinal Content Assessment Method . . . 21
3.2.1 Intestinal Content Detection 21
3.2.2 Data Partitioning 25
3.2.3 Model Evaluation 28
3.2.4 Visualisation of Results 29
3.2.5 Quantifying Detection Results 30
3.2.6 Validation in Clinical Setting 31

3.3 Approach Based on Hand-crafted Features and Classification . . . 33
3.3.1 Green-Red Colour Features . . . 34
3.3.2 Local Binary Patterns . . . 36
3.3.3 Support Vector Machine Classification . . . 39
3.4 Approach Based on Convolutional Neural Networks (CNNs) . . . 42
3.4.1 Transfer Learning . . . 43
3.4.2 Novel CNN Architecture for Intestinal Content Detection 48
3.5 Experiments and Results . . . 49
3.5.1 Data Preparation . . . 49
3.5.2 Hand-crafted Features Compared to Fine-tuning VGG architectures . . . 53
3.5.3 Novel CNN architecture . . . 55
3.5.4 Validation . . . 64
3.6 Discussion . . . 65

3.1 Background and Motivation

Since the introduction of capsule endoscopy, prevention of a clear vision of the mucosa due to intestinal content has interested researchers for different purposes. Depending on the purpose and on the available knowledge, data and computing power at the time, some of them aimed to detect specific types of intestinal content, e.g. bubbles, while others focused on detection of any type. Additionally, there are also methods that aimed to not only detect all forms of intestinal content, but to also classify each of them separately. While most methods are based on classifying entire images, some work also aimed to locate the intestinal content or determine how much of the image is covered by intestinal content. Often, such methods apply segmentation methods afterwards in an attempt to segment the area corresponding to intestinal content, despite the absence of ground truth annotations for comparison.

The earliest work on intestinal content to our knowledge, only aiming for detection of bubbles, is the work by Vilariño et al. in 2006 [91]. Their method was based on placing a threshold on the response to a bank of Gabor filters detecting the specific shape of bubbles, using an unsupervised learning method to distinguish the frames containing bubbles from the others. Wang et al [92]. recently published their work that similarly aims to detect only non-informative bubble frames based on a threshold on a filter response, but using ring shape selective filters instead. They report an improved sensitivity over Gabor filters, while achieving the same specificity on their data set. Other work that was limited to the detection of bubbles is the work by Mewes et al. [60]. They compared different descriptors based on key points detectors, testing different combinations of key points detectors and descriptors. Additionally, they extracted statistical information from selected channels of the RGB and HSV colour histograms. Their best results were obtained with the steerable filter descriptor in combination with a Hessian-affine key point detector.

Methods detecting multiple types of intestinal content did so either in multiple stages, where each stage detects a specific type of content, or detecting all types in a single stage. A method with multiple stages was proposed by Bashar et al. [4], detecting bubbles in the first stage and bile in the second stage. For the detection of bubbles Laguerre Gauss circular harmonic function filters were used, while bile was detected using colour histograms. In both stages it used support vector machines (SVM) for classification. This method was later modified by Sun et al. [84], who instead made use of local quantised histograms of classic colour local binary patterns (CLBP) for the detection

of bubbles, while it also replaced the SVM classifiers by a linear k-nearest neighbour (KNN) classifier.

Other methods proposed the detection of intestinal content in a single stage. Khun et al. [40] aimed to do so by extracting texture features in the form of colour wavelet decomposition and classifying those using an SVM classifier. Seguí et al. [77] extracted only colour information in the form of colour histograms with 64 bins, which they combined with the number of keypoints detected through speeded up robust features (SURF) keypoint detection. Apart from an SVM classifier they also separately tested a neural network (NN) classifier, generally obtaining higher accuracy for SVM.

Only a few methods have attempted to directly classify annotated regions of intestinal content instead of whole images. One of these methods is the one by Haji-Maghsoudi et al. [55] They proposed a multi-stage approach, where the first stage is aimed at classifying entire images, while the second classifies extracted non-overlapping regions of 32 x 32 pixels. In the first stage they use morphological feature extraction in combination with fuzzy k-means, which is fed to an NN classifier. The second stage segments the image with parameters based on its classification results and then extracts the regions. After extracting statistical features from those regions, they classify them through a second NN, thus obtaining classified regions.

In our case, we aim to detect intestinal content for the purpose of quantifying its presence with the ultimate purpose of comparing the patient preparation methods that were used to prepare the corresponding patients. As we discussed in the introduction of this thesis, this interests us as this problem is still unresolved in medical research, where human evaluation methods appeared to be too subjective to give conclusive results, while computerised methods employed in that context so far only considered global image features. We thus integrated intestinal content detection into an overall method for intestinal content evaluation, which combines regional results and quantifies them, converting the results non-linearly to a medical scale to provide them in an intuitive manner for medical doctors. We first present this overall method in Section 3.2, as it is independent from the chosen specific intestinal content detection, while it is necessary to understand the elements that we refer to in the chapters of each specific approach. Namely, for the detection part we investigated different approaches for the classification of regions with intestinal content, one based on traditional machine learning techniques, described in Section 3.3, and the other based on Convolutional Neural Networks (CNNs), described in Section 3.4. While one approach clearly outperforms the other in the majority of recent cases, it is interesting to compare these in the light of

the actual discussion about hand-crafted features as opposed to learnt features in relation to the theoretical basis for both.

3.2 Overall Intestinal Content Assessment Method

In this section, we detail out the different components that together compose our intestinal content evaluation method. We first explain the general framework of the regional intestinal content detection parting from CE frames as input, thus describing the elements of our intestinal content detection method that were common to both of our different regional feature extraction and classification methods. This manner is based on the classification of sub-regions of the image as we will describe in Section 3.2.1. Even though techniques based on CNNs exist that introduce for a framework that seems more efficient, namely by directly detecting and localising objects through specific CNN architectures that take entire images as input, we explain how the intestinal content we attempt to detect differs from traditional aspects and why we thus chose for a different approach. Subsequently, in Section 3.2.2 we explain how we generally partitioned the data we used to train our models into different sets taking into account the data distribution of the different training, validation and test sets to train robust models that are evaluated in an adequate manner. We then continue by describing the measures we used for model evaluation of all the models that we trained through either of the two approaches in Section 3.2.3. Afterwards, in Section 3.2.4 we explain the algorithm we used to combine the classification results in a pixel-level intestinal content detection result and how we visualised this. Finally, in Section 3.2.5 we explain how we quantified the detection results in terms of a score on a bowel cleanliness assessment scale, through a non-linear conversion, to provide results in an intuitive manner for medical doctors.

3.2.1 Intestinal Content Detection

At the core of our method of evaluating the visibility of the mucosa is the detection of the presence of substances that obstructs the view of the mucosa. This includes bile, gastrointestinal liquid, bubbles and remainders of food, which we group under the term intestinal content. An example of different types of intestinal content we aim to detect is given in Figure 3.1. Note that the presence of intestinal content is perfectly normal for a healthy intestine,

Figure 3.1: Different types of intestinal content that we aim to detect in our data set.

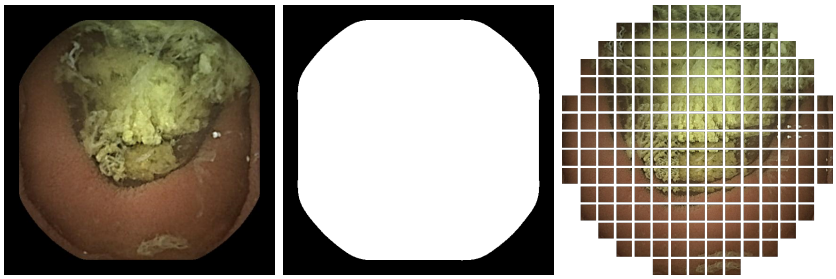
but to have a good visibility of the the intestinal wall, the mucosa, and possible pathological areas within, it is desired to limit its presence as much as possible for endoscopy procedures. This goes even more for capsule endoscopy compared to to traditional endoscopy, as we neither have active control over the capsule nor does the capsule dispose of water to clear up the view as the capsule traverses the intestine.

In essence, intestinal content detection is a form of object detection in images. However, we should take into account that the shape and characteristics of intestinal content are significantly different from those of common objects in other scenes, as intestinal content often consists of fluids that do not have a clear border and are therefore not clearly definable or detectable as a common object. This played a significant role in our considerations in the design of our detection method. While there are novel approaches that allow for accurate object detection with high computational efficiency, those are especially effective for objects with a clear outline and inadequate for the intestinal content we aim to detect. In our case of intestinal content detection, it is not straightforward to define the region we are looking to detect as a single object. Not only does intestinal content come in many different shapes and varieties, it also has vague borders that may make it hard for such algorithms to find the edges of the "object". Therefore, we instead opted for a *sliding window* detection approach, which has the additional advantage that we can fairly compare our two different approaches. We did, however, use the approach of CNNs as detectors in the detection of certain pathologies and capsule orientation estimation, in which case our detection targets did meet the requirements for those methods to be effective, and will therefore discuss them in detail in Chapter 4.

As our detection targets did not meet the requirements of traditional objects, they are both difficult for a detection algorithm and for human beings to encap-

sulate exactly within hard borders. Therefore, we considered regular sampling to be required not only for correct classification and detection performance of our method, but also for correct annotation of those samples. For this regular sampling we opted for the sliding window approach. This is a widely used term in object detection in images, referring to the way the image is processed. It generally consists in having sliding a window of a fixed size across our input image and independently process each of the sub-images we then obtain through this window. Although the idea can be used for a multitude of purposes, in object detection it is common to proceed to extract features from those sub-images and then classify those, to test whether within it we can locate the object we are looking for. There are several parameters we can vary in this procedure, such as the width and the height of the window, and the step size as we slide the window, which can be different in the horizontal than in the vertical direction. In fact, applying a convolution is also a sliding window method as we will see in Section A.3.

Other options exist for regular sampling, such selective search and region growing. However, those techniques generally use colour- or texture-based features to find regions that tightly contain an object, while in our classification procedure we designed or automatically learnt other features that do not necessarily match. Therefore, in this work, both in our traditional machine learning approach and in our CNN approach, we used the sliding window method to extract square sub-images of a regular, predefined size from the image. In the remainder of this work, we will refer to these as *patches*. In our case, we aimed for square patches as the intestinal content does not have a predetermined spread in either direction. Moreover, the images we use always contain a black frame, with a semi-hexagonal central part that shows the image actually taken by the camera, as is usually the case with capsule endoscopes of different brands. To ensure we extracted only relevant patches, i.e. patches that do not include any mask pixels, we sampled patches only from the white area in the mask image shown in Figure 3.2b. Concerning the patch size, in our experiment discussed in Section 3.5.2 we verified the optimal patch size for our models to accurately detect intestinal content to be 64×64 pixels. Using this patch size, we visualised all the patches we can thus extract from a single sample CE image in Figure 3.2. In both of our concrete detection approaches we then classified those patches at the core of our methods. Finally, from the region classification results, we extrapolated the results to the entire image by bilinearly interpolating between patches to perform the detection. In this way, we obtained final detection results as visualised in Figure 3.3.



(a) (b) (c)

Figure 3.2: (a) An example of an image from our data set. (b) The mask we apply to this image. (c) All patches we could extract from the area of interest using our method, scaled to fit into this figure.

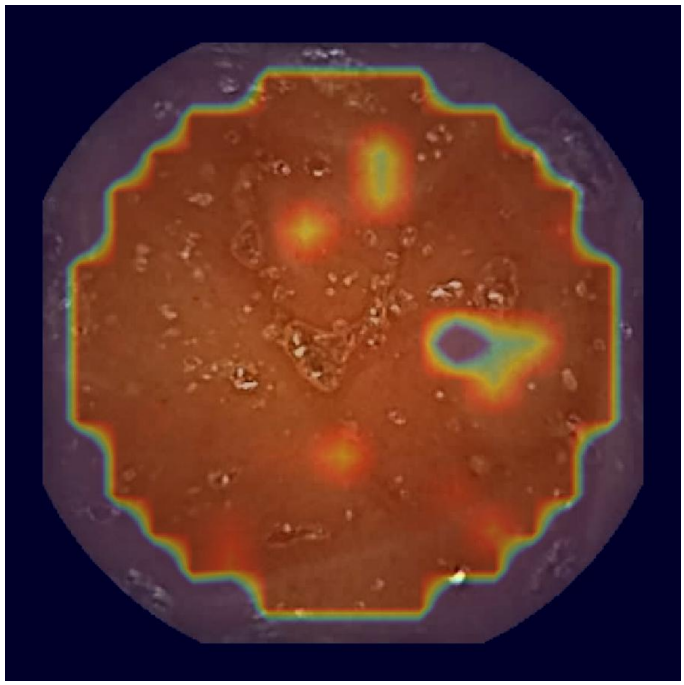


Figure 3.3: An example result of the visualisation of our intestinal content detection results.

3.2.2 Data Partitioning

To train our models in such a way that they will correctly generalise towards new images, we aimed to ensure that the data is properly partitioned into training, validation and test sets. All subsets of the partitioned data should be selected in such a way that the performance metrics are representative for entire domain of data on which our model will eventually be used. Among other things, it is important to consider variety in the input data and frequency of occurrence of the situations we attempt to classify, which can, on their turn, differ based on numerous variables. In the context of medical images, for example, age of a patient and patient history are just two examples of the numerous variables that could introduce such variety. Frequency of occurrence plays an important role when we are detecting an anomaly for example. While an anomaly on itself could be quite rare, it may be less rare in a certain setting, e.g. in a facility where only patients are redirected with very specific symptoms, or patients for who other conditions have already been ruled out by other means. There is no standard procedure to follow here, as each domain is different and comes with its own peculiarities. In our case, we are distinguishing between two classes: intestinal content (*dirty*) and the absence of intestinal content (*clean*), while we are dealing with images from many different patients. As we only have two classes, neither of which rare in the target domain and both being approximately equally frequent, we can ensure balanced sampling of both classes in our partitions without the need of excessive over- or under-sampling or data augmentation.

One problem that is important to keep in mind when training and testing a model, is that a model may *overfit* to the training samples. This means that the learning algorithm finds unnecessarily complex relations between the training samples to maximise performance on those specific samples, while they never never represent the entire output domain, along with the problem that there is often a degree of measurement error already present in the data. As these complex relations do not hold for the entire target domain, the generalisation performance becomes increasingly worse as we further overfit to the training set. Therefore, there is usually a less complex solution that, even though it may achieve perform slightly worse on the specific training samples, generalises better to new samples. While overfitting is not merely a problem of the data partitioning and it is also important to consider in the configuration of the hyperparameters of our training algorithm as we explain below, the data partitioning plays an important role. A popular method to reduce the probability that a method adjusts itself only to a specific combination of

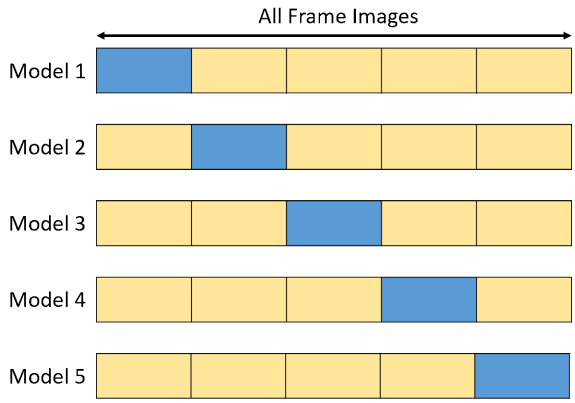


Figure 3.4: An example of 5-fold cross validation, which we used in our intestinal content classification algorithms, both in the case of traditional ML methods as in the case of CNNs. The images are thus partitioned into five different sets, each time using one different set (marked in blue) for testing and the rest (marked in yellow) for training.

training, validation and test data, is cross-validation. The technique of k-fold cross-validation involves repeating the training and testing procedures k times, using all of the available data both for training and testing, while ensuring that each image belongs to the test set exactly 1 out of k times. In our case we ensured this by first partitioning all of the available images into k subsets of equal size. Then for each time we repeated the procedure, we took one of those subsets as the test set, while we used the images from the remaining subsets for training. In Figure 3.4 we visualise this division of frame images over the different models, using $k = 5$. In this way, we thus end up with k different models, which we test individually using the corresponding test sets. Finally, to measure the overall performance of our method, it is common to take the mean performance values over all models and report those values as the performance metrics of the entire method, which is what we chose to do in our case. We applied this technique in all cases, both for our hand-crafted features models and for our CNN models.

While the above procedure essentially prevents overfitting and increases robustness with respect to the parameters that are optimised in the same procedure, in the case of our hand-crafted features models, we additionally have to deal with hyperparameters, i.e. parameters that are not derived through training and have to be set beforehand, of a separated feature extraction algorithm, as we shall explain in Section 3.3. To optimise these independently, it is

often desirable to apply the same principle as before and thus use an additional cross-validation loop to prevent overfitting of the parameter optimisation of the feature extraction algorithm, commonly looping over all possible parameter combinations. This technique of combining an *external* cross-validation loop for accurate classification evaluation with an *internal* loop for parameter adjustment is called *nested cross-validation*. Even though CNNs also have hyperparameters, in this case it has to be considered that training a CNN takes a significantly greater amount of time. At the same time, different hyperparameters affect the architecture of the network and the way it deals with the data and its optimisation with countably infinite possibilities, due to which looping over all the different possibilities is practically impossible. Instead, in CNNs researchers often tend to optimise hyperparameters empirically, namely through thoughtful experimentation and through consulting previous research on similar data sets.

This procedure is intended for testing purposes of our model in the broad sense of the word, i.e. the entire method of describing how our input data relates to the eventual prediction, while each of the model *instances* we trained throughout the procedure is referred to as a *surrogate model*. This technique is often confused with model selection. Model selection is the procedure of training substantially different models (e.g. one based on a neural network and another based on linear regression) and choosing for one or a subset of the models in the broad sense of the word, such as a single best model or a voting ensemble of different models that complement one another. This technique is mainly interesting when we have models that are trained in substantially different ways to detect different sorts of events or the same event based on different information. In the case of cross-validation, however, this is not the case, as models are optimised using the same base method, and training sets overlap significantly. The intention behind cross-validation here is thus not model selection, but model evaluation. After we have evaluated the performance of our model on the test sets that are meant to represent new data, the test sets have served their purpose and we discard them all, as selection of one of our models would likely result in overfitting of the model selection to a particular subset of our data [10]. Therefore, once we have obtained the desired statistics from our test sets and our model has been evaluated in this way, we finally train our model on all available data to obtain the model instance that we deploy into our final overall method.

3.2.3 Model Evaluation

In our case, our models are trained to classify patches into containing intestinal content (dirty) or not containing intestinal content (clean), as explained earlier. Since we have the true labels of each patch provided by our specialist(s), we can determine whether the model predicted the class correctly (true positives and true negatives) or whether it was incorrectly classified as belonging to either the positive class (false positives) or negative class (false negatives). From this information, we calculate three different metrics, the accuracy (ACC), specificity or true negative rate (SPC) and sensitivity or true positive rate (TPR):

$$\begin{aligned} \mathrm{ACC} &= \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{TN}+\mathrm{FN}}, \\ \mathrm{SPC} &= \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}, \\ \mathrm{TPR} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \end{aligned}$$

where TP, TN, FP and FN represent the numbers of true positives, true negatives, false positives and false negatives respectively. Here, the specificity would thus correspond to how many of the clean regions are correctly classified as clean, while the sensitivity would correspond to how many regions with intestinal content are correctly classified as dirty. Although these values when considered together give a good impression of the overall model performance, a desire to express the model performance in a single value has led to popularisation of the Matthews correlation coefficient (MCC). Particularly for unbalanced data sets, this value gives a better impression of the performance than the accuracy value. Namely, it is a measure that takes into account all four values of the confusion matrix, and is in essence a correlation coefficient between the observed class and the ground truth label in binary classification:

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

This causes it to have low values both in the case of high sensitivity combined with low specificity and in the case of low sensitivity combined with high specificity.

In our case of intestinal content classification, we were able to ensure balanced sampling of both classes due to the similar frequency of occurrence of the

Algorithm 1 CalculatePixelProbabilities

Input: Set of all patches P from a single image I, classified with certain probability

Input: Overlap o between patches, defined as proportion of patch size

Output: Probability image I_p where the value of each pixel corresponds to the probability

for $p \in P$ **do**
  $x_{min} \leftarrow$ minimum x-coordinate of p
  $y_{min} \leftarrow$ minimum y-coordinate of p
  $x_{max} \leftarrow$ maximum x-coordinate of p
  $y_{max} \leftarrow$ maximum y-coordinate of p
  $I_p \leftarrow$ image of same size as I, initialised with zeros
  $x \leftarrow x_{min}$
  while $x \leq x_{max}$ **do**
    $x_u \leftarrow \frac{x - x_{pl}}{width(p)}$
    $w_x \leftarrow max(0, \frac{(1 - |0.5 - x_u| - o)}{1 - o})$
    $y \leftarrow y_{min}$
    while $y \leq y_{max}$ **do**
      $y_u \leftarrow \frac{y - y_{pl}}{height(p)}$
      $w_y \leftarrow max(0, \frac{(1 - |0.5 - y_u| - o)}{1 - o})$
      $I_p[x, y] \leftarrow I_p[x, y] + \text{probability}(p) \times w_x \times w_y$
      $y \leftarrow y + 1$
    $x \leftarrow x + 1$
return I_p

different classes, due to which the accuracy value by itself is already valuable. For full transparency and completeness, however, we included sensitivity and specificity in each case, as well as single value performance metric MCC.

3.2.4 Visualisation of Results

In all of the experiments presented below, in order to determine intestinal content contamination at a pixel level from the patch detection results, we first used the overlap between patches to interpolate the detection scores at a pixel level. We did this by bilinearly interpolating the patch probabilities given by the models, assuming the given values to correspond to one of the central pixels of each patch. Concretely, we implemented Algorithm 1 to obtain the probabilities per pixel of our input image.

3.2.5 Quantifying Detection Results

The most straight-forward way to quantify the detection results, is to combine the detection results of the patches of a single image and convert them to a percentage of intestinal content over the entire image area. However, this may not be the most meaningful representation for gastroenterologists and medical researchers. Instead, it is common to categorise videos on a scale of measure. Apart from providing a more meaningful result, converting our intestinal content evaluation to such a scale could be helpful in future medical research work that may use our evaluation method as a tool or may want to compare its results side by side with human evaluation. While our method detects intestinal content at a pixel level through interpolation and can easily evaluate its presence in relation to the clean area, such as in percentages, for humans annotating or evaluating its presence at such precision is a close to impossible job.

In medical research on endoscopy and capsule endoscopy, several scales have been developed and used in the past to measure this to judge the overall cleanliness of the gastro-intestinal system of a patient during traditional endoscopy and capsule endoscopy procedures. In our work, we are not only interested in an overall cleanliness of the entire video of a patient, but we also aim to evaluate and compare results on a per-frame basis. We therefore adjusted the scale presented in the work by Beltrán et al. [5] for CE videos to apply it to frames as shown in Table 3.1, which we here denote the *cleanliness evaluation score*. In our work, we also use this scale for alternative human evaluation performed in parallel, so that the results can be meaningfully compared to those of our method.

In our method we determined the cleanliness evaluation score by categorising the video frames into one out of the four different categories of the cleanliness evaluation score based on the average probability of a pixel corresponding to intestinal content. This categorisation is not straight-forward, as we cannot assume the described cleanliness evaluation score to be linearly correlated with the amount of intestinal content present in the image. Therefore, we needed to adjust categorisation thresholds over the probabilities of pixels being dirty according to human evaluation, instead of simply placing the thresholds on equal distance from each other in linear space. Both for this purpose and for the purpose of validation discussed in the next section, we collected a set of images annotated with a cleanliness evaluation score annotations provided by two human specialists independently. They performed this evaluation independently and with a different random ordering of the images through a self-developed

Name	Perceived Level of Cleanliness
Poor	Dense intestinal content impeding evaluation of the image.
Fair	Substantial amount of intestinal content, allowing only partial evaluation.
Good	Some intestinal content, not impeding evaluation.
Excellent	No intestinal content.

Table 3.1: Description of the different categories used to evaluate the cleanliness of CE images.

tool, in which for each image they had to select the category of cleanliness they perceived out of the four different options on the scale. We then maximised the single rater reliability of the consistency-based intraclass correlation coefficient (ICC), denoted ICC(C, 1). The ICC is a statistic that allows two-way interrater reliability comparison involving multiple raters at once, while the single rater reliability indicates the reliability of the measurement if we are planning to use a single rater as the basis of the measurement [42]. The latter is exactly what we do with our method, hence the choice for ICC(C, 1). The idea of using the consistency-based variant, was to reward consistency rather than absolute values in absolute agreement, to ensure that whenever the interpretation of the amount of intestinal content differ significantly between the specialists and our method for a specific set of frames, rating consistency is still rewarded. At the same time, we aimed to reduce overfitting the ratings of our method to ratings of a single human expert that contrast with the amount of intestinal content detected by our method, and thus achieve better generalisation towards other raters and new videos.

We performed the above mentioned optimisation procedure simultaneously with the validation of our method in a clinical setting. This procedure is therefore described together with the entire validation procedure in the following subsection.

3.2.6 Validation in Clinical Setting

Finally, we aimed to validate our method in its clinical purpose, i.e. as an alternative to human evaluation in the context of cleanliness. For this purpose, we used the data we collected as described in the previous subsection, i.e. the data consisting of images annotated by experts with their perceived category of cleanliness. Similarly, we processed these images with our method, partitioning the data into two subsets and simultaneously quantifying the amount

of intestinal cleanliness on the same scale as the experts for one of the subsets of the partition and performing our validation on the other subset. As we wanted to assure that we could find thresholds that would generalise towards new videos not having participated in the optimisation procedure, we could logically no longer let those images to which the thresholds were optimised participate in the validation scores. Additionally, we needed to optimise our thresholds on a sufficient number of images to prevent overfitting towards specific cases, and at the same time guarantee certain independence from a single partition. Therefore, considering the limited total number of images available, we decided to perform this learning procedure through a randomised 5-fold cross-validation procedure on the entire set of available data. Within this procedure, we additionally ensured that the frames a single video, i.e. all images from a single patient, all ended up in the same fold, to prevent any unfair data similarity between the optimisation and validation images. For each fold, we then calculated agreement statistics between our method and the medical specialists, using the thresholds we adjusted to the images from all of the remaining folds.

The problem in determining appropriate agreement statistics, however, is that human evaluation has shown to be highly subjective, which has exactly been the main motivation to develop our method as an objective alternative. As a consequence, we do not necessarily aim for our method to adjust to any particular human cleanliness evaluation as we expect this to vary between human experts (interrater reliability) and between different ratings performed by the same expert at different times (intrarater reliability). Instead, our aim in this phase is to test whether interrater agreement between our method and human specialists is within reasonable limits of interrater agreement between human specialists only, which we discuss later in this section.

To evaluate the comparative ratings thus obtained, we calculated Cohen's weighted kappa coefficient κ_w, which is a statistic measuring interrater agreement taking into account the possibility of agreement occurring by chance [15]. Apart from correction for an estimation of agreement occurring by chance as the standard kappa coefficient, this weighted version of Cohen's kappa also allows us to merge both full and partial agreement into a single value between each two raters, along with an estimation of its 95% confidence interval. We chose to use kappa with linear weights, denoted κ_1, instead of quadratic ones, since the choice for quadratic weights tends to systematically give higher values than linear weights [93], thus having the undesirable effect of decreasing differences between raters in our situation. The magnitude of agreement for different values of κ_w has been reason for discussion. While different inter-

pretations of values in different ranges have been suggested in literature, the truth is that its value also depends on other factors than agreement, such as prevalence and bias [79], while the tendency of weighted kappa to yield consistently higher values further complicates correct interpretation of its magnitude. Therefore, here we mainly use the values for interrater comparison between the different pairs of raters without using such tables in an attempt to interpret their magnitude. In the absence of ratings from more specialists, it is also difficult to deduce any conclusions with statistical significance about the interrater agreement of our method with humans in general. Therefore, to assess whether our interrater agreement with each of the specialists individually that is within reasonable limits of interhuman agreement, we calculated κ_1 both separately over each of the different folds and over the concatenated results from all folds, in order to obtain a series of interrater agreement values that can give a more detailed picture of the agreement ranges or variances of κ_1 along with its confidence intervals over different sets of videos and overall. Finally, we calculated the single rater reliability of the two-way mixed, absolute agreement-based ICC, denoted ICC(A, 1), over the concatenated results from all folds, as κ_1 does not allow for assessment of more than two raters at the same time. This could correct for the influence of bias and prevalence κ_1 in our two-way comparisons. We calculated this statistic twice, namely once for all of our three raters and once leaving our method out, to assess the impact of our method on the resulting values and the corresponding confidence intervals.

3.3 Approach Based on Hand-crafted Features and Classification

When we started our work, traditional machine learning techniques were still widely used, although CNNs had already started to outdo traditional methods at the time in the field of biomedical computer vision. However, at that moment we did not have the facilities at our disposal for training such networks, while the theory behind CNNs was still not as widely accepted and understood at the time as it is today. Additionally, we believe that insights and techniques from the traditional fields of image processing and machine learning are also useful in both the understanding and the application of modern end-to-end learning.

Hand-crafted features are intuitive and therefore inviting to work with. With this term of hand-crafted features, we refer to features that are extracted

through hand-crafted algorithms, i.e. algorithms developed to always extract the information somehow representing the same information, be it texture, colour, shape or other. Even though the amount of available data plays a role both for hand-crafted and learnt features, as we shall see, in the case of learnt features we need significantly less data as hand-crafted features are already specific and even in the case we want to optimise certain parameters of the feature descriptor, this number does not compare to the number of parameters to be learnt in the case of learnt features discussed later, having a significantly reduced solution space.

In the case of hand-crafted features in the medical field, one generally discusses with medical specialists about the features they observe in images from the problem at hand when they distinguish between one case and another, to try to find feature descriptors that aim to extract exactly this information. In our case, we closely cooperated with medical specialists from Hospital La Fe in Valencia to understand the different types of intestinal content, the characteristic features of each and finally, what feature descriptors we could use to distinguish between these. We noticed that there was an important difference in both colour and texture features of normal mucosa compared to intestinal content and therefore based the design of our feature extractor on this. Separately extracting colour features and texture features and merging those into a single feature vector, we finally extracted a feature descriptor of 32 values, with 2 values corresponding to colour features and 30 to texture.

3.3.1 Green-Red Colour Features

Colour features are those features extracted directly from the colour values that represent an image, without considering relative ordering of pixels. Here we can think of descriptors such as mean colour values, max and min colour values or colour histograms. In the case of our images collected from the capsule endoscope, images are represented in the RGB colour space as described in Section 2.3, in such way that for every pixel in the image, we have a 3-tuple (R, G, B), in which the values are given for the red, green and blue channel respectively. However, many other colour spaces exist, such as HSV, HSB, CYMK and CIE-Lab. Another representation of colours may make colours that are relevant to a specific problem at hand better distinguishable, due to which it may be helpful to first convert our pixel values to that colour space.

For the features we extracted for intestinal content detection in our work, we focused on the differences in green and red colour tones between the two

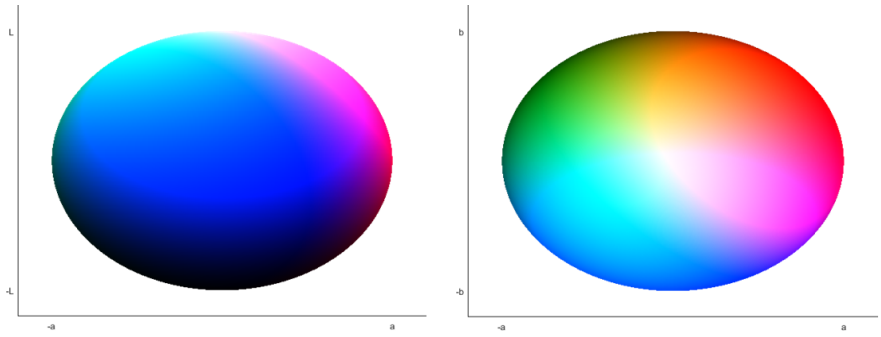


Figure 3.5: The sphere that results from plotting the different combinations of the L, a and b values from the CIE-Lab colour space.

areas. Namely, clearly visible mucosa and its pathologies often have a pink to reddish colour tone, while intestinal content is often dark yellow to green. White is also a colour we often see for intestinal content in the reflection and refraction of the capsule's light on bubbles that are closely grouped together, while white may also show in the reflection of light from the mucosa when it is lightly humid. In either case, we found that closely grouped bubbles are more clearly distinguishable for their characteristic texture rather than colour, as we describe later.

We therefore searched for a colour space that emphasises differences between red colour tones characterising well-visible mucosa along with potential pathological areas, and the greenish colour tones that characterise intestinal content. Through research and experiments, we found the CIE-Lab colour space to be the most suitable, as not only is colour information separated from lightness and saturation in this colour space, but it is also conceptually related to the opponent colour theory in the a and b channels [94] and should thus be more discriminative between green and red colour tones. To visualise the properties of this representation, Figure 3.5 shows different views of the sphere that shows the resulting colours from different combinations of values of the L, a and b components. As can be seen in the figure, the L-component controls the amount of white, the a-component controls the amount of green and the b-component controls the amount of blue. Since a and b represent the different colour tones, whereas the L-component mainly controls the intensity, we discarded the L-component in our work and calculated the average values of the a- and b-components for inclusion in our final feature vector.

3.3.2 Local Binary Patterns

For the extraction of textural information to distinguish between intestinal content and clean mucosa, we used local binary patterns (LBP). The classic method of LBP [63] works as follows for grey (intensity) images. Since we used colour images, where each pixel is defined as a combination of three colour channels (red (R), green (G) and blue (B)), we first created three different intensity images per original image: one with all the R values, one with all the G values and one with all the B values. We then applied local binary patterns to each of these images separately, which can then be considered intensity images for a certain colour, while we later concatenated the results. In general, in LBP we construct an LBP image where each pixel is a representation of the local texture. These values are obtained by comparing the intensity value of the corresponding pixel with those in its neighbourhood and assign a unique numeric value for each situation of interest. From this binary image, we can then construct histograms to represent the general texture of the whole input image or image regions.

To explain how we calculate the LBP values, we will part from an example of an intensity image given in Figure 3.6. We then independently compare the value of each pixel of our input image to the value of each of the pixels in the neighbourhood, as defined by the LBP parameters, individually. In our example we show the case of comparing a single pixel, which is the central picture in Figure 3.6. While the radius r and the number of neighbours to take into account P are two important parameters in LBP, in our example we will explain the case where $r = 1$ and $P = 8$, which corresponds to exactly all the pixels that directly surround the pixel of interest. From this comparison, we construct a binary number, in which each bit corresponds to exactly one of the adjacent pixels and represents how it compares to the central pixel: it is on if its value is equal to or greater than the value of the central pixel, while it is off if it is smaller, as shown in the upper part of Figure 3.7.

The resulting binary number thus represents the texture information of the neighbourhood of the central pixel. For convenience, we then convert it to a decimal number as shown in the bottom part of Figure 3.7. The resulting decimal number for the pixel is then assigned to the pixel at the same position in our resulting LBP image, as visualised in Figure 3.8, thus obtaining a resulting image where the value at each pixel essentially indicates the texture we have found at that location. Afterwards, we take the histogram of the LBP image in order to obtain the feature vector.

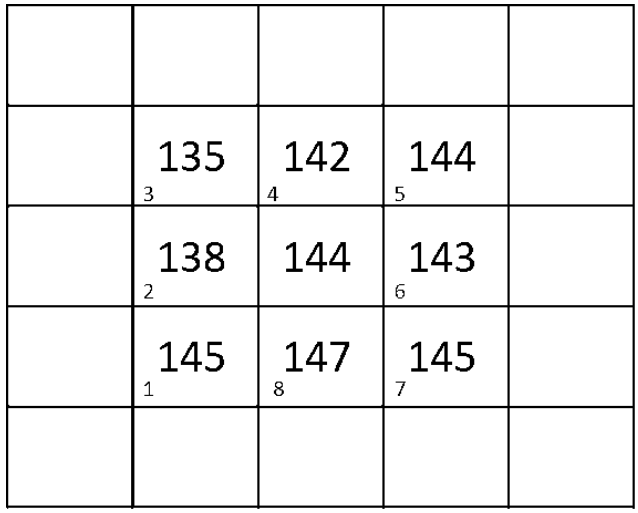


Figure 3.6: The intensity values of a pixel of an image and its neighbours.

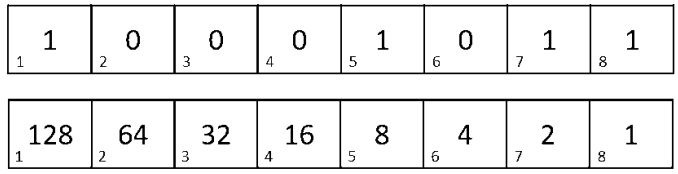


Figure 3.7: The construction of the binary number for the central pixel, with the multiplication values for the conversion to a decimal number and the resulting calculation of the conversion.

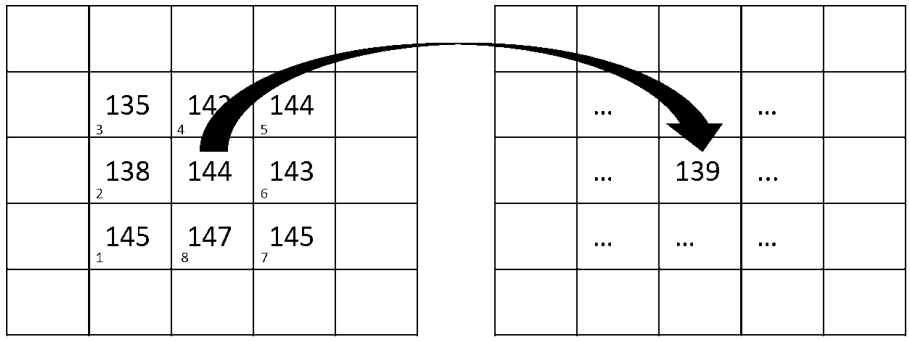


Figure 3.8: The calculated LBP value for the central pixel, which takes the position of the central pixel in the LBP image.

While this general form of LBP is a useful feature extractor, it has been shown that is not always descriptive enough. Particularly, it was noticed that some patterns are more discriminative in images than others, the so-called *uniform* patterns. These are the patterns that, when considering the binary string to be circular, have at most two changes from 0 to 1 or the other way round. In other words, these are all the patterns where all bits corresponding to the same value are connected in a circular manner, thus essentially consisting of a series of 0 values and 1 values. The method of uniform local binary pattern then assigns a single, identical value to all patterns that have more than two changes between 0 and 1. All the uniform patterns are instead assigned a unique value. Since there are $\binom{P}{2}$ patterns with a first change from 0 to 1 and a second change from 1 to 0 in a circular fashion, and an equal number of patterns with a first change from 1 to 0 and a second one the other way round, while there are two patterns that have no change, consisting of either only values of 1 or only values of 0, and finally a single value for all of the remaining patterns, uniform LBP has a total of $\binom{P}{2} \times 2 + 2 + 1 = 59$ values.

Separately, a rotation invariant version of LBP, $\mathrm{LBP}^{\mathrm{ri}}_{P,R}$ was developed. This variant considered that the same pattern in the image, when rotated, was assigned a different value by the definition of LBP. Therefore, this variant aimed to align the values that detect the same pattern in a rotated form and assign to them an identical value. Concretely, the binary LBP number is circularly right shifted bit-wise until the maximal number of most significant bits are 0. In case $P = 8$, for example, it has been shown that we can obtain 36 different values in this way.

From the combination of these two variants, finally the variant was derived that we use in this work, denoted $\mathrm{LBP}^{\mathrm{riu2}}_{P,R}$ [62]. This variant is invariant under rotation and robust against resolution changes. As in uniform LBP, we only assign unique values to the uniform patterns, and assign the same value to all other ones. But as in $\mathrm{LBP}^{\mathrm{ri}}_{P,R}$, we first perform a circular right bit-wise shift until we obtain the number with the maximal number of 0-valued most significant bits. Concretely, if the number of changes (from one to zero or zero to one) in our binary number is at most 2, we rotate the number until it has the greatest number of adjacent zeroes in the first positions. To each of these situations we then assign a unique value, while all other patterns are represented by a single default value. As all of the 0-valued bits are always connected, we thus end up with $P + 1$ distinct values from 0 to P for the uniform patterns, as shown in Figure 3.9, plus 1 additional default value $P + 1$ that is assigned to all of the remaining patterns. This finally results in $P + 2$ distinct possible values for the $\mathrm{LBP}^{\mathrm{riu2}}_{P,R}$ method.

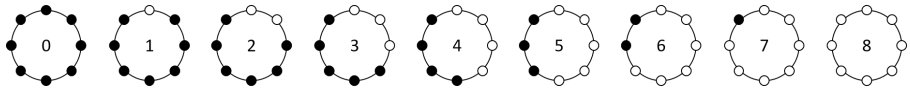


Figure 3.9: The 9 distinct uniform LBP patterns in the rotation-invariant $\text{LBP}^{\text{riu2}}_{8,R}$, with their corresponding LBP value from 0 to 8 in the middle. The final LBP value in this method is the value $P + 1$, 9 in this case, assigned to all of the remaining (non-uniform) patterns, yielding $P + 2$ different values in total.

In our case, we use the $\text{LBP}^{\text{riu2}}_{P,R}$ variant for our intestinal content feature extraction. However, instead of applying them to a greyscale image, we apply them to each of the R, G and B colour channels of the image individually, after which we calculate the histograms for each channel separately and finally concatenate them to obtain our final feature vector. This way of using LBP over RGB images is indicated by the prefix RGB, i.e. $\text{RGB-LBP}^{\text{riu2}}_{P,R}$. In this way, we thus obtain a feature vector of $3(P + 2)$ values for our texture feature descriptor. Combined with the colour features, this yields a final feature vector of $3(P + 2) + 2$ values.

3.3.3 Support Vector Machine Classification

After thus extracting features from our CE image patches, we need to generate a model that is capable of classifying new images, i.e. to determine whether it contains intestinal content or not. For this purpose, we need to train a classification algorithm capable of constructing such a model, using the features extracted from the training data as explained before, along with the training data labels. Classification algorithms work by applying mathematical techniques to find an distinguishing border between the samples of the two classes according to certain conditions. Many such algorithms have been developed over the years. In our work, we choose to use one of the currently most popular ones in the state of the art which has shown to be effective for our type of problem, namely Support Vector Machines (SVM).

SVM classifies data by finding the hyperplane that best separates the data points of one class from the data points of the other. Our feature vector is in a significantly higher dimensionality, namely in $\mathbb{R}^{3(P+2)+2}$ with $P \geq 8$, due to which it is impossible to simply visualise the data. For the purpose of explaining the method, however, here we will imagine our data as being two-dimensional and plot their locations in $\mathbb{R}^2$ in Figure 3.10. In this figure, as is the general case in $\mathbb{R}^2$, the hyperplane SVM searches for is a straight line. We can imagine how this process scales to higher dimensions: for three dimensions

we would end up with a plane instead of a line, while for higher dimensions we would end up with a hyperplane.

Often, different possibilities exist for choosing the straight line that distinguishes the training samples of both classes, as shown in Figure 3.11. In this case, we choose the line which has the greatest margin between the two classes. The margin is the maximum width of the area between two parallel lines at equal distance from the the border for which this area does not contain any data points. This distance is visualised by the blue line in Figure 3.12, while the parallel lines are visualised as dashed lines. The area corresponding to the margin is marked in yellow. The vectors on the boundary of this area (thus on the parallel lines in the image) are called the support vectors, which the name of the method is derived from and which have an important role in the calculation of the border. They are marked by a red border in Figure 3.12.

In reality, samples can often not simply be separated by a linear hyperplane as in the example above. For those cases, the kernel trick has been invented. [6] The kernel trick involves a mapping of the original data points to a new space, with the goal of rearranging the points to make them better separable by a linear hyperplane. The function that is used for this mapping is called the kernel function. The kernel function often has parameters for which we need to find the optimal values. In our method, we use a Gaussian kernel function which has a parameter γ that correlates to the size of the Gaussian kernel.

Formally, in SVM classification, we attempt to find an optimal solution to the

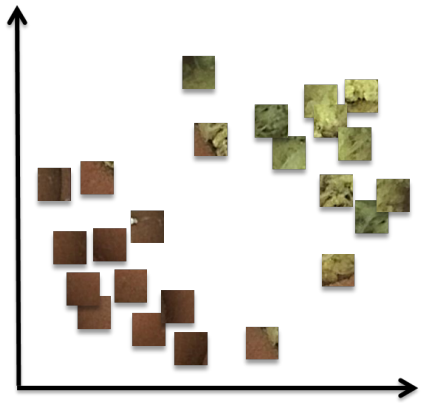


Figure 3.10: An example of a hypothetical plot of our CE intestinal content patches represented by 2-dimensional feature data.

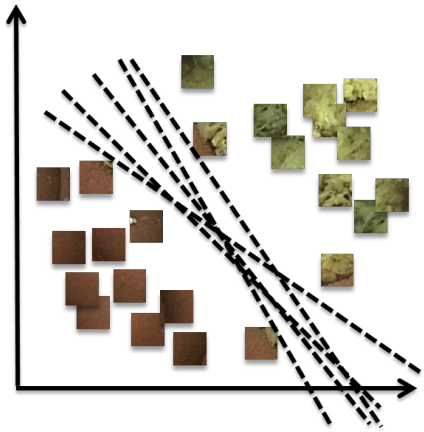


Figure 3.11: Possibilities for different lines that would separate patches with intestinal content from those without.

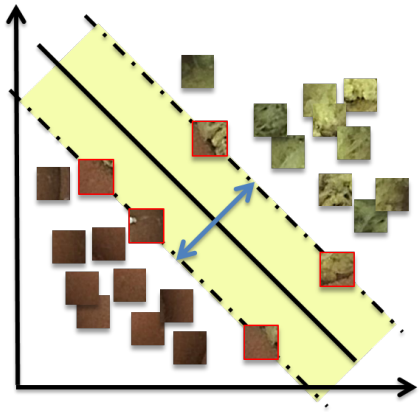


Figure 3.12: The optimal border found by the SVM algorithm, with the support vectors marked by a red border, the margin represented by a blue arrow and the area corresponding to the margin marked in yellow.

problem

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned} \tag{3.1}$$

where $\mathbf{w}$ is a normal vector to the plane that defines the margin, b is the offset of the plane, C defines a cost parameter defined by the user, ξ_i defines a slack variable to soften the constraint on x_i, i.e. that determines to what extent it is allowed to deviate, and ϕ is a transformation function that transforms its input vector to a new space [17]. The function $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ is then defined as the kernel function, which in our case is the is the radial basis function (RBF), given by $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$ with $\gamma > 0$. The parameters C are often optimised through a grid search, as we describe for our case in Section 3.5.2.

3.4 Approach Based on Convolutional Neural Networks (CNNs)

Our next experiments were based on the end-to-end learning domain using CNNs. In this domain, not only is a classifier trained to automatically distinguish between features of different classes, but the very extraction of the features themselves is learnt simultaneously in an artificial neural network. In contrast to the approach of the previous chapter, the selection of features is thus no longer the responsibility of the researcher. Here instead, the art of the researcher boils down to investigating the best network architecture or new architectural elements or a suitable combination of elements from different architecture, which are capable of efficiently extracting and combining the valuable information for the problem at hand from the input data. Additionally, the data selection and partitioning, along with input and output scaling, become ever more important. The approach of CNNs has become so widely popular, that we did not want to include the general theory behind CNNs in the main text. Therefore, we refer the reader who is not (sufficiently) familiar with the concept to Appendix A, where we describe the fundamentals of the theory of CNNs in detail.

Both as a proof of concept and as to leverage the advantageous availability of existing pretrained standard CNN models, our first experiments were per-

formed using fine-tuning of pretrained CNN architectures. This is part of the domain of transfer learning with CNNs, where information of previously trained models on more generic image domains or other specific domains is used in the specific problem to be solved by the researcher. As the result of this experiment deemed the CNN approach successful for our problem, we decided to further investigate this approach to find a solution that could further improve performance. Therefore, in this section we first discuss transfer learning in Section 3.4.1, and discuss our final approach of designing and training our own CNN architecture in Section 3.4.2.

3.4.1 Transfer Learning

Training a CNN from scratch may require a significant amount of labelled input data and computational resources, depending on the problem domain. Still, CNNs have been successfully applied in different areas of computer vision in recent years. This is not only due to the availability of more advanced computer resources in recent years, which is not available to everyone, but mainly due to the concept of *transfer learning*. Transfer learning refers to any usage of existing neural network architectures, of which have been pretrained on vast amounts of general labelled image data, e.g. the popular ImageNet database [19]. As the lower layers are more relevant to the specific problem as explained above, only those layers and their corresponding weights are highly specific to the problem at hand. One way to benefit from this is to remove the last classifying layer from the pretrained network and use the remaining network as feature extractor. Another way is to keep the weights of the highest layers fixed and only refine the weights of the lowest layer(s) using our specific training data as input. This process is referred to as *fine-tuning*. The first available pretrained CNN structure was AlexNet from 2012, in which convolution operations were repeated several times before applying a pooling layer, with the idea of obtaining more complex features at every spatial scale [43]. In the following years researchers extended this work, largely inspired by the yearly ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [74], mainly focusing on increasing the depth of the network.

For the purpose of transfer learning, the popularity of CNN architectures for classification is often directly related to their performance in the ILSVRC challenge. The purpose of this challenge is to evaluate algorithms for object detection and image classification at large scale [74]. While this challenge was at the origin of transfer learning, the original motivations behind the challenge were to compare progress in detection across a wider variety of objects, where

ImageNet took responsibility for the expensive part of labelling images, and to measure the progress of computer vision for large scale image indexing for retrieval and annotation. With the first edition of this challenge being organised in 2010, only by the 2012 ILSVRC edition *deep learning* (learning through deep ANNs) gained significant popularity in Computer Vision research, when that edition was won by the CNN architecture that is now commonly known as AlexNet.

Out of all the architectures available, we experimented with VGG, Inception and ResNet in our work on intestinal content detection. These are the classification architectures that at the time were among the highest-ranked on the aforementioned challenge at the time, while they each have substantially distinctive elements that could make a difference on certain data sets. We discuss each of these and their distinctive elements in detail below. In our initial experiments, however, we saw that we obtained significantly better results for VGG than for the other ones, which is why we devote most attention to this architecture.

3.4.1.1 AlexNet

The AlexNet architecture consists of five convolutional layers and three fully-connected layers. The first and second convolutional layers are followed by a normalisation layer and a max-pooling layer, while the fifth convolutional layer is also followed by a max-pooling layer without normalisation. All of the pooling layers in the AlexNet architecture use overlapping pooling, with a stride of 2 and a receptive field of 3. Peculiar about their original training procedure was that, to paralellise the training procedure, it was trained on 2 GPU's, where the kernels of each layer were equally divided over the GPU's, but the third convolutional layer as well as all the fully-connected layers performed their computations over all the results of the previous layers by having the GPU read the memory of both itself and the other GPU.

3.4.1.2 GoogLeNet or Inception

While in most previous architectures the focus was on making networks deeper, Inception (initially named GoogLeNet [85]) introduced a concept to make the network wider and deeper at the same time, while avoiding to make it computationally more expensive by implementing dimensionality reduction. It made the network wider in the sense that it implemented the multi-scale idea, using

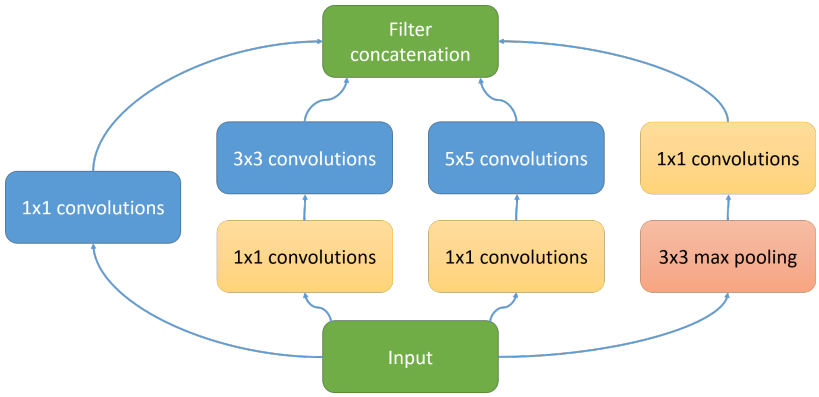


Figure 3.13: The Inception Module as originally implemented in GoogLeNet/Inception. This module combines different filter sizes and max pooling in parallel, using a 1×1 convolution beforehand to reduce the dimensions and thus the number of parameters of the network.

convolutions with different filter sizes in the same layer. By doing so, it also attempted to approximate sparsity in CNN layers, even though, by convention, the components it is made up from, are in fact dense, while clustering neurons that fire together, after [3] and intuitions of the Hebbian principle, i.e. "neurons that fire together, wire together". This architecture won the image classification challenge of ILSVRC 2014 [74].

All of these principles are essentially implemented in a module that was named the *Inception module*. This module groups 1×1, 3×3 and 5×5 convolutions, all applied to the same input at the same level in parallel, as well as a max-pooling unit, after which the results are concatenated. The authors indicate that these filter sizes were chosen mainly for convenience, as they together guarantee no issues with pixel alignment. This module is visualised in Figure 3.13. In this figure, we can see another interesting concept of this network: the implementation of 1×1 convolutions between the input from the previous layer and the convolution units, where the number of filters in this layer is reduced. Not only does this significantly reduce the number of parameters by effectively compressing the input combining the information cross-channel, but it also introduces an additional non-linearity, as they gave it a ReLU activation function. While many other architectures require an important effort in experimenting with different filter sizes in different layers, with the Inception module, researchers commonly need to invest their time in this experimentation, as the most common convolutional filter sizes are thus implemented in parallel within the architecture.

After the first version, newer versions of Inception have been developed that each time improved the classification performance of the initial architecture. Even though the latest version of Inception is already version 4, the aforementioned concept, that all versions consist in, are that they are largely based on the Inception module. The architectures mainly differ in the number of layers and the number of Inception modules used within.

3.4.1.3 ResNet

The final architecture we discuss here is, ResNet [33], developed by Microsoft. ResNet implements the concept of *residual connections*, which it derives its name from. With this, it attempts to deal with one of the most common problems in deep learning, which is that of vanishing gradients. As researchers attempt to make architectures ever deeper in an attempt to learn more complex features, they become more prone to vanishing gradients as we have to deal with further repeated multiplications in back-propagation that can make the gradients infinitesimally small. This led to deeper architectures often performing worse and set a introduced a limit on the number of layers up to which architectures would improve. Although researchers proposed different ideas to deal with this problem, none of them seemed to offer a generic solution, until the proposal of residual connections. The idea of residual connections seems simple: it introduces a parameterless identity shortcut connection from one layer to a deeper layer, which is then summed to that layer's output, skipping one more or layers in between. The *residual block* that ResNet introduced is visualised in Figure 3.14. However, this concept ensures that as we stack more layers, we will never perform worse than for fewer layers, as the identity mappings would essentially reduce the effective layer size. At the same time, the authors argue that it is easier to learn a residual mapping with respect to an earlier feature map instead of a complete mapping, which is essentially the type of mapping the shortcut connection forces the layers in between to fit. It has to be noted that the concept of residual connection was not new as it was already introduced before in Highway Network, of which ResNet is essentially a specific case. Nonetheless, Highway Net generally achieves worse performance and does not generally tend to converge to the specific case of ResNet, which may be due to its significantly increased solution space as a more generic architecture.

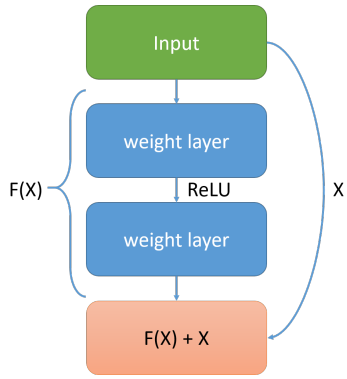


Figure 3.14: The residual block introduced in the ResNet architecture.

3.4.1.4 VGG

VGGNet, or simply VGG, was the first architecture to consistently use 3x3 kernel sizes in the convolutional layers, replacing larger kernels employed in AlexNet. By concatenating multiple convolutional layers with this kernel size, the network becomes deeper and more complex features can be represented at a lower cost. Until today, these architectures are the most widely used purely sequential architectures for classification. Being fully sequential as well as having consistent filter sizes in the convolutional layers, makes them intuitive architectures to comprehend and to fine-tune. VGG-16 and VGG-19 are the two different architectures its authors proposed, differing in the number of convolutional layers they contain [80]. Although we experimented with the other architectures listed in this chapter, these two VGG variants were the architectures we found to be most effective in our case and the ones we therefore eventually used in our work. Both architectures are visualised and compared in Figure 3.15. They both consist of 5 sequential blocks, which, in turn, consist of sequential convolutional layers followed by a max pooling layer. Although we replaced the classification layers of the architecture with our own tailored to our problem, both originally have three fully connected layers and a soft-max layer for classification, leading to a total of 16 and 19 *weight layers* respectively.

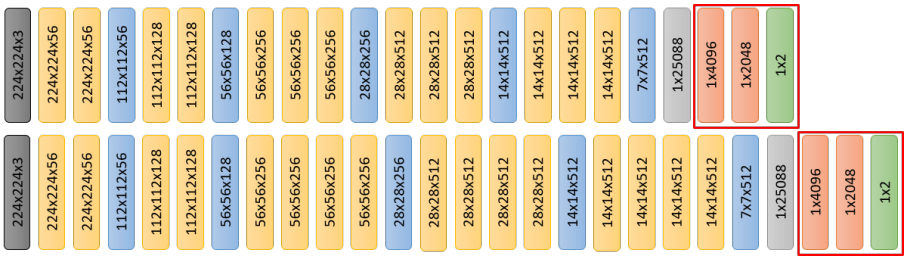


Figure 3.15: An overview of the VGG-architectures we implemented, with our modified VGG-16 at the top and our modified VGG-19 at the bottom, where each rectangle represents a layer of the network. The layers surrounded by the red rectangles are the layers we defined by which we replaced the fully connected layers of the original VGG-16 and VGG-19 architectures. The text corresponds to the output sizes. The far left layers are the input layers, yellow are convolutional layers, blue are max pooling layers, light gray are flattening layers, red are fully connected layers and green are the softmax classification layers.

3.4.2 Novel CNN Architecture for Intestinal Content Detection

In our final approach, we designed, implemented and trained a novel CNN architecture, with the objective of achieving a light-weight model with a lower number of parameters than our previous classification methods, resulting in lower prediction times and a limited usage of resources, without sacrificing the high accuracy obtained in the previous experiment. In terms of initial network depth and stride, this architecture was inspired by the architecture presented for CE images by Jia et al. [37], for which the authors obtained a high accuracy in bleeding detection in CE images.

The proposed architecture is shown in Figure 3.16. The base model of this architecture consists of 4 blocks, of which the first starts with two convolutional layers with a Leaky ReLU activation function and batch normalisation, while the other blocks each have only one such layer. Each of these block ends with another convolutional layer with Leaky ReLU activation and batch normalisation, but with a stride of 2, which we verified to result in higher test classification accuracy in our case than a max pooling layer in its place, as also shown in [82]. The top model consists of flatten layers to reshape the three-dimensional output from base model of the network to a one-dimensional vector, after which two fully connected layers perform the classification. The first of these layers consists of 128 neurons with a LeakyReLU activation function and batch normalisation. The last classifying layer consists of 2 neurons

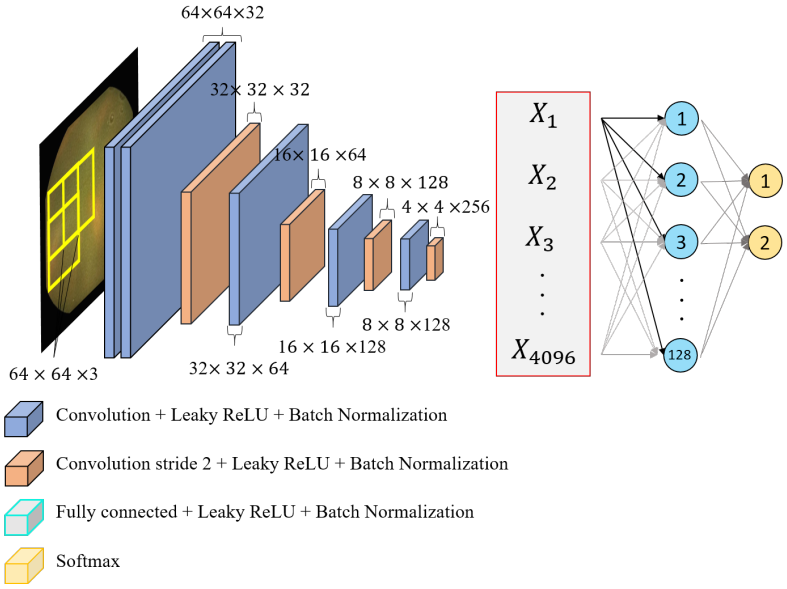


Figure 3.16: Visualisation of the CNN architecture proposed for intestinal content classification in this work.

with the softmax activation function, resulting in outputs that correspond to the probability of the input belonging to either one of our two classes. We also experimented with the use of drop-out, but we found that it did not have any impact on the results for our data when using batch normalisation, while using drop-out instead of batch normalisation led to worse results.

3.5 Experiments and Results

3.5.1 Data Preparation

3.5.1.1 Collection

The data we used for all the experiments in this section was collected at *Hospital Universitari i Politècnic La Fe* from Valencia, with previous approval of the study from its Ethics Committee of Clinical Research and with written informed consent from all subjects involved. The study was carried out in accordance with Spanish law for the protection of human subjects. For the

training procedure of our models we collected relevant segments of 35 different videos of capsule endoscopy procedures, performed with the PillCam SB 3 model. An expert gastroenterologist extracted the parts corresponding to the small intestine through their official RAPID Reader software, capable of reading the proprietary file format of the video recordings, from which we subsequently systematically extracted all frames at regular intervals of one minute. This led to a total of 563 individual frames of 576×576 pixels. For the validation in a clinical setting, we collected 30 additional videos from new capsule endoscopy procedures. From these videos we extracted frames exactly the same way, leading to a total of 854 additional frames of also 576×576 pixels for the validation procedure.

3.5.1.2 Conditioning

Since we attempt not only to detect but also to locate and quantify the intestinal content, at the core of our method we use a model based on classification of regions as opposed to entire images. For this purpose, we developed an annotation tool that lets the annotator select a video from a list, after which it loads the corresponding frames that can be processed sequentially or accessed directly through another list. The tool predivides the area of the image that corresponds to the recording, ignoring the black frame, into patches of 64×64 pixels, with a step size of 32 pixels both in the horizontal and vertical direction, thus allowing an overlap of half in both dimensions, which we first verified to be adequate for our case through experiments. When the annotator clicks anywhere in the image, the tool finds the centre of the closest predefined patch and marks this as dirty or clean, visualised with a red or green border around the patch respectively, depending on the mouse button used for the click. The patch can then be cleared by clicking it another time. In this way, two specialists selected those patches that either consisted completely of intestinal content (*dirty*) or were completely void of intestinal content (*clean*). This tool is shown in Figure 3.17 with an example annotation of both dirty and clean patches performed with the same criteria we used throughout our annotation procedure. We decided for such patch-based annotation because intestinal content can have unclear borders and an irregular spread, while patch-based annotation allows us to obtain annotations as detailed as required and still allows us to capture sufficient information about its spread and transition to normal mucosa, without having to quantify the areas. It also prevents the annotator from having to make doubtful decisions about where exactly the border is, as areas along the border do not necessarily need to be annotated in

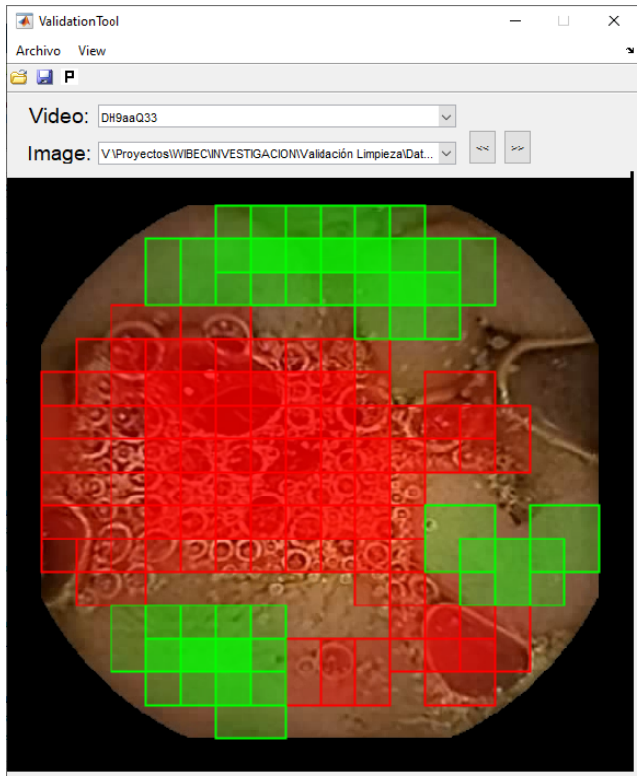


Figure 3.17: The annotation tool we developed for our specialists to annotate the collected data. The visualised annotation is performed by the same criteria our experts were instructed to use throughout the annotation procedure.

this way. Importantly, we instructed the specialists to select any encountered pathological areas naturally present in the included frames, and classify them as clean. In this way, we ended up with 26746 clean patches and 28547 dirty ones. Hereafter we refer to this data set as the *training set*.

For the validation of our method in a clinical setting, we aimed to compare the cleanliness evaluation of our method with the independent per-image evaluation by two medical specialists. For this purpose, each of the 854 images from the 30 additional videos were annotated by two specialists with their perceived cleanliness evaluation score, as we explain in detail in Section 3.5.4. In the remainder of this section we refer to this set with its corresponding cleanliness score annotations from both specialists as the *validation set*.

Fold	Training		Validation		Test	
	Dirty	Clean	Dirty	Clean	Dirty	Clean
1	13579	13902	3395	3476	4541	4823
2	13086	14343	3272	3586	4971	4555
3	13063	14310	3266	3578	5283	4809
4	13206	14255	3302	3564	5077	4888
5	14018	13667	3504	3417	3911	4162

Table 3.2: Number of patches per fold in which we partitioned our training set for the 5-fold cross-validation in the training procedures. All final and surrogate CNN models were trained and evaluated using these same partitions for training, validation and testing.

3.5.1.3 Partitioning

To avoid overfitting to our training data and to improve the robustness of our method, we performed 5-fold cross-validation on the training set in the procedure of training of our models. Additionally, to perform fair evaluation of the performance of each model, we ensured that the patches from the test sets in these cross-validation train/test splits always originated from different images than the patches in the corresponding training sets. To achieve this, we first shuffled the entire set of images in the training set and divided them into five subsets or folds of images of equal size. For each fold, we then let the patches annotated in the corresponding images be our test patches, while the patches extracted from the images in all of the remaining folds served as our training patches in that case, thus creating five pairs of training sets and test sets of patches. This is visualised in Figure 3.18, where the green images represent the test images and thus correspond to a different fold each time, while the blue images represent the images from the remaining folds. While the data was originally sufficiently balanced, we still ended up with imbalanced data in some of our created subsets due to varying number of annotated patches over the different images. Therefore, in the experiment with our novel CNN architecture described in Section 3.5.3, we chose to apply undersampling where necessary. Concretely, we randomly removed patches of the majority class in each of our training and test sets if the corresponding number of images was more than 10% greater than the number of of the minority class, to reduce this difference to exactly 10%. We then split each of the training sets into training and validation sets using an 80%-20% split. In this way, we ended up with the final sets of patch images of the sizes given in Table 3.2, which is the data we used for training and evaluation of all methods implemented and compared in the first two experiments.

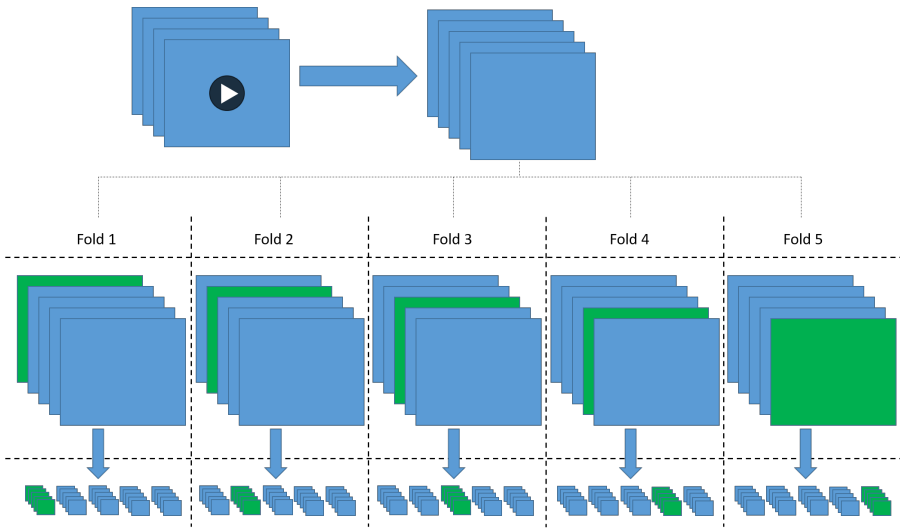


Figure 3.18: The process of our data collection and partitioning. From videos we extracted frame images in which patches were annotated by specialists. We then partitioned the images into five folds (green), while we let the remaining images be the training images in those folds. From these, we extracted the annotated patches into equivalent folds, so that we ended up with the corresponding five sets of training (blue) and test (green) patches.

3.5.1.4 Public Availability

Public data is scarce in the field of capsule endoscopy. This is even more the case for images with annotations provided by clinical experts. Therefore, we decided to publish our data online exactly as we used them for training our final algorithms as described in this section. Namely, the patched labelled CE images used for training and evaluating the models can be found at `https://cvblab.synology.me/PublicDatabases/CECleanliness/CECleanlinessTraining.zip`, while the CE images used for the validation in a clinical setting can be found at `https://cvblab.synology.me/PublicDatabases/CECleanliness/CECleanlinessValidation.zip`.

3.5.2 Hand-crafted Features Compared to Fine-tuning VGG architectures

In this experiment, we compared the fine-tuning of the popular VGG architectures with hand-crafted features. For the hand-crafted features, we defined

a descriptor consisting of both colour and texture information, and trained a classifier on those features. Specifically, our feature vector was the concatenation of uniform rotation-invariant local binary patterns (LBP^{riu2}) [62] on the R, G and B colour channels and the mean of the a and b colour channels of the patch image in the $L^*a^*b^*$ (CIELAB) colour space [58]. We then trained an SVM classifier on the extracted feature data to construct a model capable of distinguishing between both classes. This method is hereafter referred to as Feature Extraction-SVM (FE-SVM).

For the optimisation of the SVM parameters C in Eq. (3.1) and γ in the kernel function, we performed an extensive grid search using internal 4-fold cross validation to prevent overfitting, thus training 4 times for each possible combination of parameters, each time leaving out a different partition of the data for validation. First, we performed a coarse grid search attempting all combinations of γ and C with γ ranging from 2^{-15} to 2^3 and C ranging from 2^{-5} to 2^{15}, with the exponent increasing in steps of 2 for each parameter. We then performed a fine grid search in which each of the two parameters takes values between 2^{k-1} and 2^{k+1}, with k being the exponent with highest average validation accuracy for that parameter among the 4 folds in the coarse grid search. Here we increased k in steps of 0.25 instead and we chose the value for which we obtain highest average validation accuracy among the 4 folds.

In the same way, we simultaneously optimized the parameters of LBP^{riu2}, concretely the number of neighbours (8 or 16) and the radius (ranging from 1 to 10), along with the patch size of s x s pixels ($s = 64$, $s = 32$ and $s = 16$), attempting each different combination of both these parameters and the SVM parameters. As for the patch size, the patches were annotated in the greatest size $s = 64$ in order to programmatically extract a smaller region around the centre of each patch as our data for the smaller patch sizes, as all patches were either entirely clean or entirely dirty. Figure 3.2c shows an example of all patches we can extract from an image this way when $s = 64$.

We compare our traditional machine learning method with a method based on deep learning through convolutional neural networks (CNNs) using transfer learning as explained in Section 3.4.1, more specifically, fine-tuning. In our case, we separately fine-tuned two VGGNet architectures, namely VGG-16 and VGG-19 architectures. VGGNet was the first architecture to consistently use 3x3 kernel sizes in the convolutional layers, replacing larger kernels employed in AlexNet. By concatenating multiple convolutional layers with this kernel size, the network becomes deeper and more complex features can be represented at a lower cost. We chose for these architectures since the VGGNet architectures achieved highest accuracy in previous work on CE images [49].

Additionally, they are the most widely used purely sequential architectures and have consistent filter sizes in the convolutional layers. For these reasons they are intuitive architectures to comprehend and to fine-tune.

For training our models, we made use of the Keras framework in Python with TensorFlow as its backend. We retrained the weights of our modified VGG-16 and VGG-19 architectures from the lowest layer of the highest convolutional block in both cases. We partitioned each of our training data sets into a training set and a validation set, ensuring a 20% of the data in the validation set. As per the image size used to pretrain the networks, we had to rescale our patch images from 64x64 to 224x224. We set all the other parameters for the training procedure empirically. We used the accuracy as our performance metric and trained the networks for a maximum of 100 epochs or until validation accuracy no longer improved during 15 epochs or more. We set the batch size to 4 due to memory limitations and the optimization algorithm to SGD. For SGD we used the binary cross-entropy loss function and set the momentum to 0.98, the learning rate to 0.0001 and the decay to 0.0. To increase the robustness of our models, we also used data augmentation with a variation of up to 2% in zoom level, rotations and flipping operations.

For FE-SVM, we first optimized the parameters of our feature extraction in a cross-validation procedure as explained in Section 3.2.2. The optimal parameters we found in this way were: 16 neighbours p and a radius r of 1 for LBP, and 64 for s in the patch size $s \times s$. For the CNNs we used the optimal value of 64 for s in the patch size $s \times s$ found in FE-SVM. We did not need to optimize any feature extraction parameters for this method, as the CNN learns the features by itself. We used 5-fold cross-validation in both methods. The results obtained in this procedure per fold are given in Table 3.3, both for FE-SVM using the optimal parameters and for the different CNN architectures. Visualising the probabilities as a heat map as explained in Section 3.2.4, using the VGG-19 model for which we obtained highest accuracy, we obtained images as shown in Figure 3.19.

3.5.3 Novel CNN architecture

In this experiment, we compared the performance of our architecture with the separately implemented base models of the architecture by Jia et al. [37] and of the popular VGG-16 architecture [80], both trained under equal conditions as our proposed method, i.e. exactly the same data and partitioning of those as described above, same hyperparameters and without post- or preprocessing, to

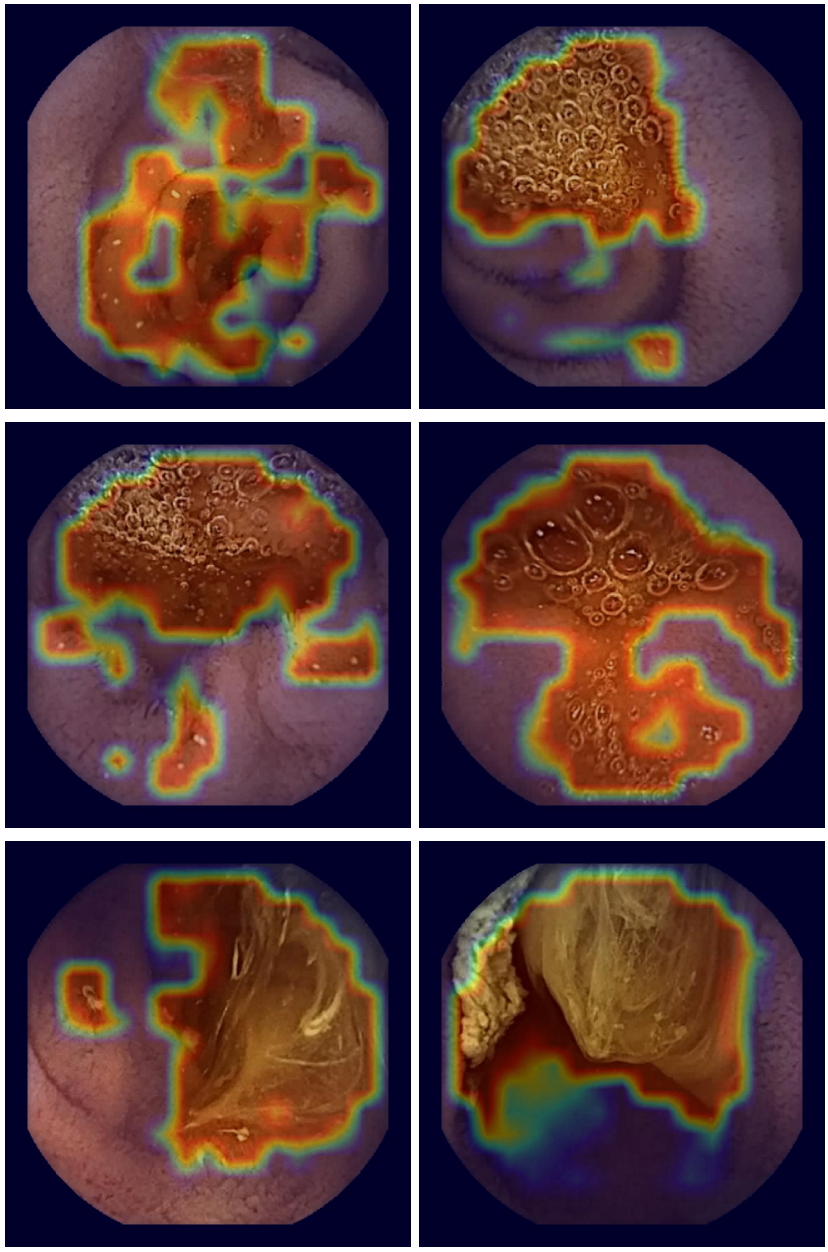


Figure 3.19: Visual results of our detection method using the VGG-19 model interpolating the probabilities per pixel from the patch probabilities and displaying as a heat map (continued on the following page).

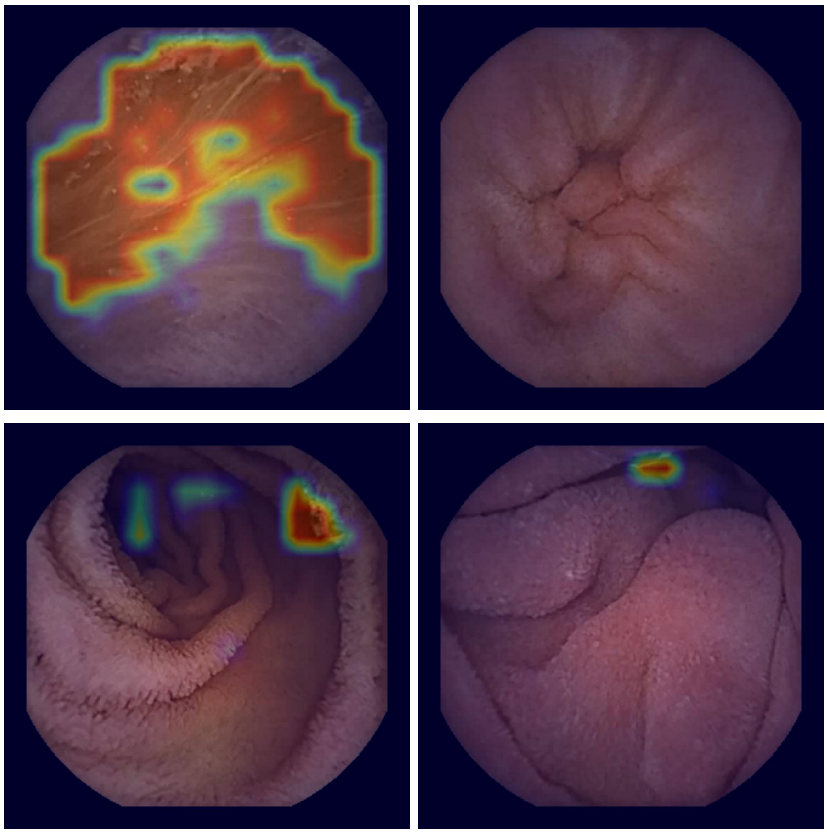


Figure 3.19 (cont.): Visual results of our detection method using the VGG-19 model interpolating the probabilities per pixel from the patch probabilities and displaying as a heat map (continued). Red areas indicate intestinal content detections, while blue areas were classified as clean mucosa.

ensure a fair comparison of the classification performance of each method. In assessing the performance of our method, we specifically compared our method with VGG-16 because it performed comparably well to VGG-19 in our previous experiment, while having significantly fewer parameters. Additionally, we used the same top-model for all architectures, which is described below.

We set the values of the hyperparameters for the CNN training procedure empirically. We eventually used a batch size of 16 and a learning rate of 0.0005. For VGG we used SGD as an optimiser, as it outperformed nadam, with a momentum of 0.9. For the other models we used nadam. We did not use any data augmentation and did not rescale the input images or preprocess

Table 3.3: Results obtained in terms of accuracy, sensitivity and specificity.

Method	Fold	Accuracy	Sensitivity	Specificity
FE + SVM	Fold 1	85.32%	89.43%	80.42%
	Fold 2	87.62%	86.94%	88.21%
	Fold 3	86.84%	89.83%	84.12%
	Fold 4	87.83%	93.64%	82.23%
	Fold 5	88.75%	89.60%	87.50%
	Average	**87.32**%	**90.73**%	**83.71**%
VGG-16	Fold 1	93.49%	92.59%	94.64%
	Fold 2	95.04%	93.50%	96.45%
	Fold 3	94.10%	90.83%	97.53%
	Fold 4	93.71%	89.46%	98.73%
	Fold 5	95.32%	94.98%	95.86%
	Average	**94.33**%	**92.27**%	**96.61**%
VGG-19	Fold 1	95.15%	94.59%	95.84%
	Fold 2	95.92%	95.43%	96.35%
	Fold 3	94.33%	91.00%	97.83%
	Fold 4	94.62%	91.75%	97.77%
	Fold 5	95.74%	96.60%	94.48%
	Average	**95.15**%	**93.87**%	**96.45**%

them in any other way; we directly used the normalised RGB colour channels of the patch images as extracted from the CE images, resulting in input vectors of size $64 \times 64 \times 3$ for all of our models.

Additionally, for VGG-16 we leveraged the advantage of the availability of existing weights pretrained on the ImageNet data set to instantiate the weights. Namely, it has significantly more parameters than the other two methods due to which we cannot provide a fair comparison training it from scratch on our limited data set, which would also not be the common use case considering the average size of CE data sets. Subsequently we trained the model in two stages, adjusting the randomly initialised model head in the first stage, while we unfroze all layers and trained the entire network in the second stage.

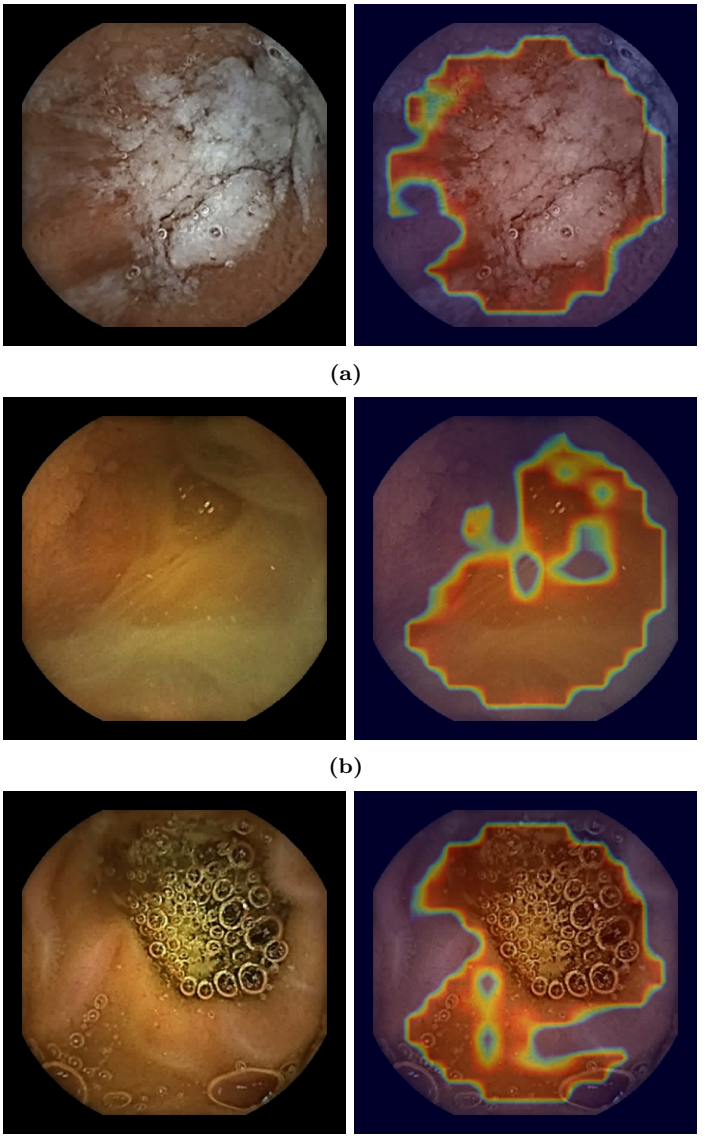
In the classification of dirty and clean patches, the results we obtained for each of the models trained over the different folds are given in Table 3.4 for all of the aforementioned methods. The proposed architecture obtained highest accuracy, followed by VGG-16 and finally by the architecture proposed by Jia et al.. We also evaluated the size of each model in terms of number of parameters and storage space, along with the average time required for a prediction of a batch of 2000 patch images, measured over the processing of

Method	Fold	Accuracy	Sensitivity	Specificity	MCC
VGG-16	1	94.27%	96.81%	91.57%	0.8861
	2	94.68%	92.29%	96.86%	0.8939
	3	94.35%	98.65%	90.44%	0.8906
	4	95.02%	98.10%	92.06%	0.9023
	5	95.66%	96.03%	95.27%	0.9132
	Average	**94.80%**	**96.38%**	**93.24%**	**0.8972**
Jia et al.	1	93.40%	97.26%	89.30%	0.8701
	2	93.46%	92.54%	94.31%	0.8689
	3	93.84%	97.40%	90.59%	0.8793
	4	93.96%	96.87%	91.16%	0.8808
	5	94.96%	96.06%	93.79%	0.8992
	Average	**93.92%**	**96.03%**	**91.83%**	**0.8797**
Proposed	1	94.96%	95.92%	93.94%	0.8992
	2	95.67%	94.84%	96.44%	0.9134
	3	95.05%	98.09%	92.28%	0.9028
	4	94.77%	96.64%	92.97%	0.8961
	5	95.71%	95.41%	96.04%	0.9143
	Average	**95.23%**	**96.18%**	**94.33%**	**0.9051**

Table 3.4: Results for the different methods over each of the different folds, with the average for each method in the bottom row.

100 of such batches on an NVIDIA GeForce GTX 950 GPU using a MATLAB 2018a implementation. Here we obtained lowest values for the model by Jia et al., with a prediction time of 0.33 seconds, 520,800 parameters and 4 MB of storage space on disk. This was followed by our method, with 0.50 seconds prediction time, 1,708,610 parameters and 20 MB storage space. Finally, VGG-16 obtained the highest values with prediction time of 0.81 seconds, 14,977,730 parameters and 117 MB of storage space. Eventually, integrating the model of the proposed architecture into our method of calculating the probabilities per pixel, we obtained images as shown in Figure 3.20.

We separately analysed the detection results on the images that contained pathologies in our test set, which were thus not used for training the corresponding model as described in Section 3.5.1.3. With the help of one of our experts, we identified 10 images in this set showing clear pathologies. Within these, our expert diagnosed 2 cases of active bleedings, 3 of ulcers, 5 of angioectasia and 1 of a polyp. Through manual analysis of the intestinal content detection results with the proposed method over these images, we found that one of the ulcers was partly detected as intestinal content, while all of the remaining pathologies were correctly classified as clean intestine by our method.

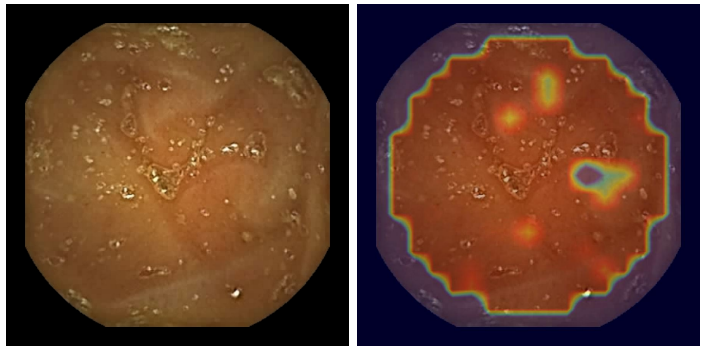


(a)

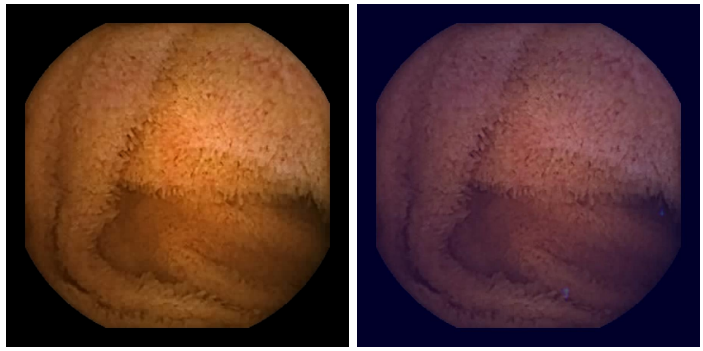
(b)

(c)

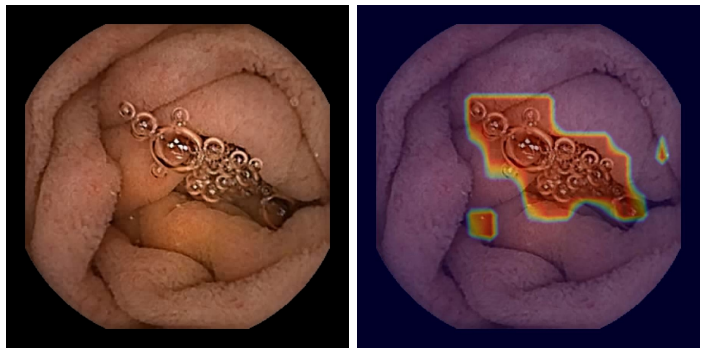
Figure 3.20: Visual results of our intestinal content detection method based on a novel CNN architecture, with interpolated pixel probabilities overlayed as a heatmap as before (continued on the following page).



(d)

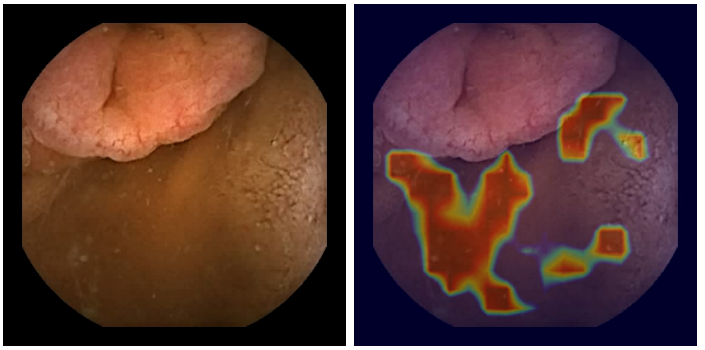


(e)

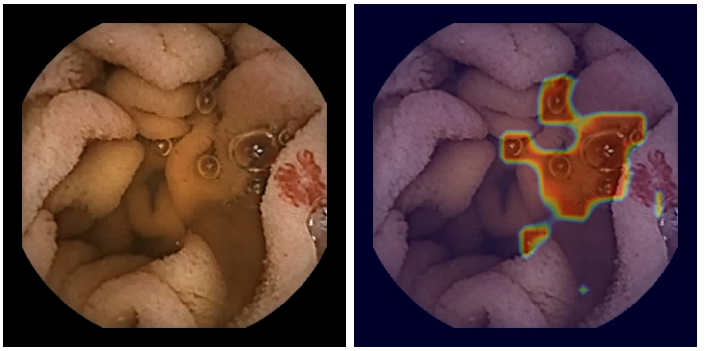


(f)

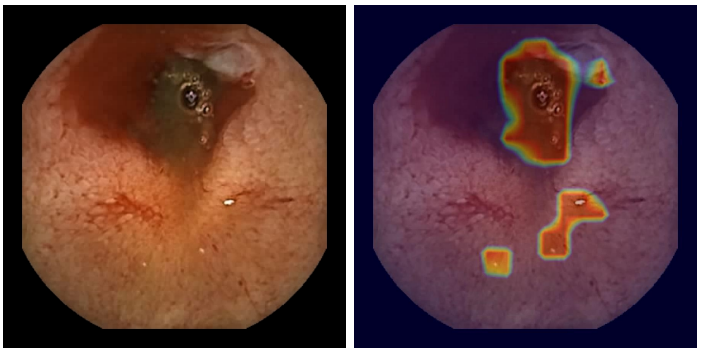
Figure 3.20 (cont.): Visual results of our intestinal content detection method based on a novel CNN architecture, with interpolated pixel probabilities overlayed as a heatmap as before (continued on the following page).



(g)

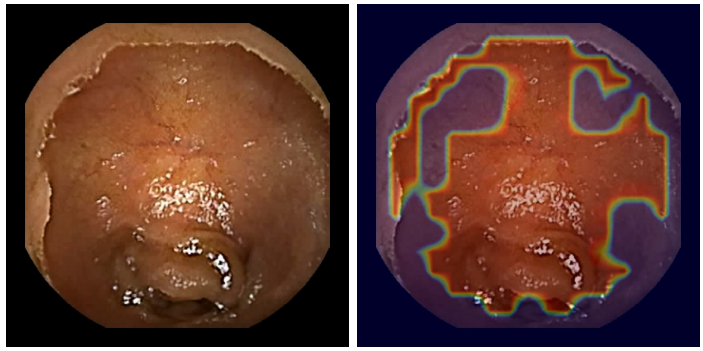


(h)

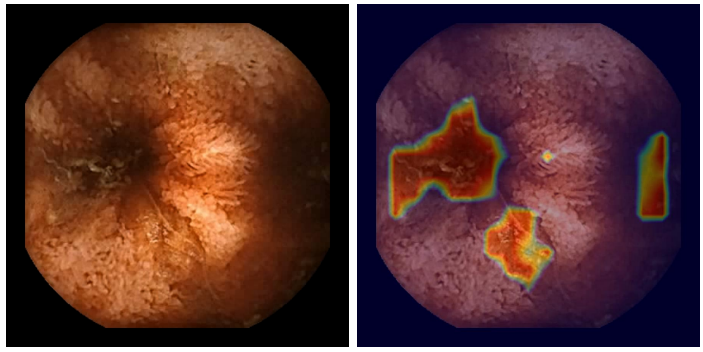


(i)

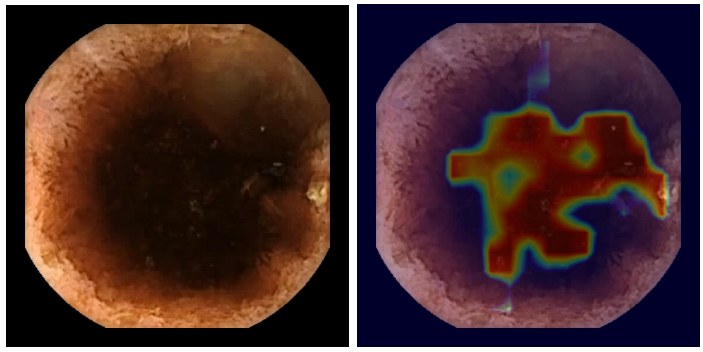
Figure 3.20 (cont.): Visual results of our intestinal content detection method based on a novel CNN architecture, with interpolated pixel probabilities overlayed as a heatmap as before (continued on the following page).



(j)



(k)



(l)

Figure 3.20 (cont.): Visual results of our intestinal content detection method based on a novel CNN architecture, with interpolated pixel probabilities overlayed as a heatmap as before (continued). Figures g, h and i are a part of the test set of our model. The remaining images all come from the validation set, which was not used for the training procedure of our method. Figures j, k and l show characteristic examples for which there was a significant difference between the assigned evaluation score and the evaluation scores assigned by both human specialists.

The case of the polyp is shown in Figure 3.20g, a case of angioectasia in Figure 3.20h and the case of ulcer that was partly detected as intestinal content in Figure 3.20i, with the bleeding classified as clean intestine in the same image.

3.5.4 Validation

After testing the intestinal content detection performance of our method in the previous phase, in this experiment our objective was to test the clinical validity of our method as described in Section 3.2.6. As explained there, we performed this procedure simultaneously with the optimisation of the categorisation thresholds, through a 5-fold randomised cross-validation procedure in which we ensured that all images from a single patient always ended up in the same, single fold to prevent any leakage of input data from the optimisation set to the validation. In this way, each image is used for validation exactly once, and per image we thus obtained a single non-optimised cleanliness evaluation score for validation purposes, having adjusted the corresponding thresholds for that fold only to the remaining images. We then measured and compared the selected agreement statistics both in a per-fold and overall fashion, the latter of which obtained by aggregating all of the per-fold evaluation scores.

In order to convert the classification of patches to an evaluation score for a frame of a CE video, we needed to determine the thresholds for the categorisation in the way we explained in Section 3.2.6. The threshold values over the averaged detected pixel probabilities of intestinal content that we found to be optimal for assessing the videos in a way similar to human assessment, reported by their means and standard deviations over the five different folds of our validation set, were 0.42 ± 0.022 between Excellent and Good, 0.66 ± 0.017 between Good and Fair and 0.94 ± 0.022 between Fair and Poor. Using these thresholds for the images in each corresponding fold, we obtained per-fold cleanliness evaluation scores.

For the validation of our method, we calculated the interrater statistics that allow us to validate our method as described in Section 3.2.6. On a per-fold basis, we calculated κ_1 for each pair of different raters, allowing us to validate our method through comparison with the evaluation by human specialists. We used a MATLAB implementation of weighted kappa [8] to perform this calculation. The resulting κ_1-values and their corresponding 95%-confidence intervals are given in Figure 3.21. Except for the case of fold 2, the values found for our method were always within the 95% confidence interval of the κ_1 between the human specialists. Overall, concatenating the ratings over all

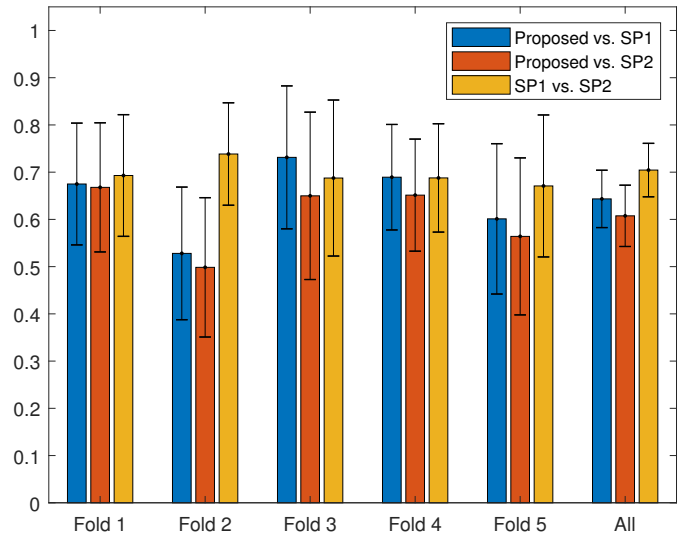


Figure 3.21: The agreement values of κ_1 measured over the different folds and over the concatenated results of all folds, each plotted with its corresponding 95% confidence interval.

folds, we measured a κ_1 of 0.643 and 0.608 for our method (Proposed) with specialist 1 (SP1) and specialist 2 (SP2) respectively, while we measured a κ_1 of 0.704 for the agreement of both specialists with each other, with confidence intervals of $(0.583, 0.704)$, $(0.543, 0.672)$ and $(0.648, 0.761)$ respectively. Finally, using SPSS v25.0 we calculated ICC(A, 1) both over only the ratings of the human specialists and over all raters at once, to measure its change with the exclusion or inclusion of our method respectively. Over the two human raters we thus calculated an ICC(A, 1) of 0.817 with a confidence interval of $(0.745, 0.864)$, while including our method among the raters ICC(A, 1) was 0.770 with a confidence interval of $(0.739, 0.798)$.

3.6 Discussion

For our main objective, we performed several sequential experiments to achieve a fully automated procedure for the assessment of the visibility of the mucosa in CE videos, iteratively attempting to get closer to our objective of developing a method that could be used in clinical practice.

Our first experiment was focused on the comparison two different methods to automatically detect regions of intestinal content in CE videos: one based on hand-crafted feature extraction and SVM classification and the other based on CNNs. While previous work has applied complex, multi-stage approaches, often involving hand-crafted features, in this work we showed that similarly high accuracy can be achieved with the decision of a single model, without any preprocessing or postprocessing of the images, obtaining an accuracy of 95.15%. The results of this experiment helped us to determine to go with CNNs to develop the classifier we finally wanted to implement in our evaluation method. The VGG architectures, however, are still relatively expensive, as the processing of a batch of 2000 image patches still took 0.81 seconds with VGG-16.

Therefore, in the next experiment we decided to focus on a light-weight, self-designed architecture and test whether with such an architecture, trained from scratch instead of fine-tuned, we could achieve higher efficiency without sacrificing classification accuracy. While in this next experiment we aimed to design a CNN architecture that would approach the performance of VGG-16 with significantly fewer parameters and a significantly reduced prediction time for faster processing of videos, our results show that with the proposed architecture we can even improve classification performance over VGG-16, despite having fewer than 2 million parameters as opposed to nearly 15 million for VGG-16 and requiring only 62% of the prediction time of VGG-16. Compared with the architecture proposed by Jia et al., we obtained significantly higher accuracy for the proposed architecture, at the cost of more parameters and a 52% increased average prediction time. Additionally, through the inclusion of images that showed pathologies, our method correctly recognised bleeding, angioectasia, a polyp and most of the ulcers as a part of the clean intestine. Although this looks promising, it will be necessary to test this on a greater set of pathological images in the future before we can draw any conclusions. Of our test cases, only one ulcer appeared to be partly detected as intestinal content. For ulcers, the situation is more complicated than for the other pathologies, as they are often accompanied by intestinal content in the case of perforated ulcers, while in other cases they can still be difficult to distinguish from intestinal content in still image frames even for human experts. In the case of Figure 3.20i, for example, the part of the ulcer detected as intestinal content has very similar characteristics to some of the intestinal content in Figure 3.20a, while it also has bubbles in its vicinity.

Finally, in our third experiment, the validation of our method in a clinical setting, we found that the agreement between our method and each of the

specialists individually over all images approached the interhuman agreement, yet the values fell just outside of the interhuman 95% confidence interval. Especially in the case of fold 2 our method performed significantly worse than in other folds, scoring a lower agreement with either one of the human specialists than the lower limit of the 95% confidence interval of the κ_1 of the two specialists with each other, while in all of the other folds it was always within that interval, achieving higher agreement with SP1 for fold 3 and fold 4. Observing the case of fold 2 in greater detail, we noticed that one of the videos in this fold contained a significant number of frames on which bleeding could be observed, where our method assigned evaluation scores of excellent cleanliness, while both specialists agreed on poor cleanliness. Measuring the ICC(A, 1) both with our method included and with our method excluded, we found that the resulting value of the single rater reliability remained within the 95% confidence interval of the single rater reliability of ICC(A, 1) with our method excluded. Although we cannot prove significance as discussed earlier, we believe that this information in combination with the comparative values found for κ_1 does give a good indication that our method's agreement with the human specialists is within reasonable limits of inter-human agreement.

It is important note that even though the specialists performed the validation procedure independently, they work together in the analysis of CE procedures on a regular basis, which may make them more prone to agreeing with each other in their perception of cleanliness than what would be the case between specialists from different institutions. We also highlight that, when we reviewed the frames with highest disagreement between our method and both human specialists, in many cases both of them no longer agreed with the evaluation score they assigned in first instance. This demonstrates the lower intrarater reliability of human evaluation due to the earlier discussed subjectivity, while intrarater reliability is always 1 in the case of our method, as it will consistently assign the exact same evaluation scores to each image regardless of the number of repetitions.

Through reviewing the mentioned frames, we identified three specific cases for which the evaluation by our method significantly differed from human evaluation. j, k and l Examples of these are shown in Figure 3.20j, Figure 3.20k and Figure 3.20l. In the first case, shown in Figure 3.20j, the capsule around the camera appears to be caught in a bubble itself. As our method has been trained to detect bubbles, it may detect the smooth surface of the mucosa without its characteristic texture, while it may also detect that the light emitted by the capsule is reflected by the surface of the bubble. We argue that it is a difficult for our method to distinguish between the case of the capsule

being inside a greater bubble, allowing us to observe the mucosa nearly perfectly, and the case of smaller bubbles on the surface of the mucosa that reflect the light much more rigorously and thus do not allow for correct observation of the mucosa behind it. However, although the general occurrence of such frames is relatively limited, we believe that in future work we could address this issue by actively collecting more such images to include in our training set. The second type of images with a significant difference images that show bleeding, as shown in Figure 3.20k, which were numerous in one of the videos in Fold 2. In these images, the blood is not detected as intestinal content by our method, which is in fact desired behaviour as the scale addresses the remainders of food, bile and intestinal liquid. However, our specialists did assign a low evaluation score in the presence of bleeding. We argue that this may be an effect of human psychology, assigning a bad score to an image in the presence of diseases despite evaluating other factors, which could show an important advantage of our method in its objectiveness. Finally, our method sometimes detects sparse intestinal content in the lumen hole where human observers do not, as shown in Figure 3.20l. Even though it is not a false detection, it can be argued that, as the mucosa in the darkness cannot be observed either way, it does not impede visibility. In future work, it would be interesting to investigate how the relative location of intestinal content, e.g. against the background of the lumen hole or against the background of the near intestinal wall, influences the cleanliness in a frame as perceived by humans. The presence of intestinal content against the background of the lumen hole does, however, provide information about the preparation of the intestine that we can expect to normalise over the course of an entire video.

Considering the limited number of specialists available for the cleanliness evaluation of the images in our validation set, we were forced to use part of these images, annotated with both assigned evaluation scores, for adjusting the categorisation thresholds. This is a limitation of our method, as it could cause our method to over-adjust to the specific annotators. Ideally, we would like the scores used to adjust the thresholds of our method and the scores used to validate the method to be completely independent from each other, e.g. originating from different observers, as this would show that our thresholds generalise well to other observers. We would like to revisit this in our future work when we could obtain more data from different clinical centres around the world. We do, however, emphasise that we did guarantee independence between validation and adjustment of the thresholds in terms of our data, as we not only used different videos for each of those, but additionally obtained entirely new videos for the validation procedure, ensuring that none of the videos used for validation was also used in the training procedure.

Chapter 4

Determination of Capsule Orientation for Motility Analysis

In this chapter, we describe the methods we use for determination of the capsule orientation, which is built upon R-CNN-based techniques, and how we use this to create an approach to to visualise intestinal motility, through approximating a traditional intestinal manometry. At the end of the chapter, we describe the individual experiments in detail and discuss the results.

Contents

4.1 Background and Motivation 70
4.2 Tunnel Detection 72
4.3 R-CNN Fundamentals 74
4.3.1 Introduction 74
4.3.2 Informed Region of Interest Proposals 79
4.3.3 RoI Pooling 81
4.3.4 Bounding Box Anchors 84
4.3.5 Bounding Box Regression 86
4.3.6 Loss Functions 88
4.3.7 Model Evaluation 89
4.4 Proposal-based Detectors 90
4.5 Regression-based Detectors 92
4.6 You Only Look Once (v3) 95
4.7 Experiments and Results 96
4.7.1 Data Preparation 96
4.7.2 Tunnel Detection 99
4.7.3 Optical Manometry Approximation 101
4.8 Discussion 108

4.1 Background and Motivation

Important current issues with capsule endoscopy compared to traditional endoscopy surround the fact that the capsule cannot be actively controlled. Namely, in traditional endoscopy a gastroenterologist actively controls the endoscope, which allows him or her to keep track of the location and orientation of the camera in real-time. In capsule endoscopy, however, it is difficult to determine the location and orientation of the capsule at a specific moment of the procedure. Work has been done in an attempt to solve this shortcoming, for example by investigating motion between subsequent frames. However, in our own experiments with motion analysis on real videos of entire capsule endoscopy procedures, we have observed this to be difficult, due to the complexity of aligning frames with the combination of a relatively low frame rate and the significant forces that act upon the capsule due to the intestinal motility.

The latter mentioned intestinal motility has been subject of research only to a small extent in the field of capsule endoscopy, even though it is an important factor both both in capsule motion and in bowel disease. Namely, according to certain epidemiologic reports, only in the United States as many as 30 million people have intestinal motility disorders. Additionally, available data from the medical literature indicate that worldwide over a quarter of all gastrointestinal conditions can be contributed to intestinal motility disorders [56]. Symptoms differ depending on the affected part of the GI system and the underlying cause, but include gastroesophageal reflux disease, gas, severe constipation, diarrhoea, abdominal pain, vomiting, and bloating [25]. These disorders often occur by themselves, but can also be secondary to a malignant condition. While they can occur in any part of the gastrointestinal tract, here we focus on the motility of the small intestine, as this is the trajectory covered by CE.

Although different methods have been used in the past, the method that is widely accepted as most reliable for diagnosis of motility disorders of the small intestine nowadays is intestinal manometry. This is a procedure which is generally performed through placing a thin, pressure-sensitive tube into the upper gastrointestinal tract to measure muscle contractions and identify spasms. Placement is done through traditional endoscopy, for which the patient needs to be sedated, as discussed in chapter 2. An overview of intestinal manometry, visualisations of the recordings that inspired our approximations introduced in this chapter, along with a discussion of alternative methods was published by Hansen [31]. Although it is well-tolerated by patients, it is an invasive

procedure which requires thorough interaction from and analysis by a gastroenterologists.

While some studies attempt to use CE for intestinal motility analysis to avoid an invasive manometry, mainly through measuring overall transit times, an interesting new method that attempts to alleviate the invasiveness is a one based on a new endoscopic pill, namely the wireless motility capsule (WMC). This capsule concurrently measures pH, temperature, and pressure [23]. However, this capsule is does not have a camera on board, due to which it has to be specifically used for intestinal motility analysis. Often, it would instead be useful to derive information alongside a visual examination, to avoid having to choose between different procedures, or even having to do both.

Although work focused on deducing intestinal motility information from CE video is scarce, one interesting study we came across attempted to automatically detect and annotate contractions in CE videos through detection of tunnel frames and classifying series of 9 consecutive frames into contractions and non-contractions, using extraction of hand-crafted features, such as the co-occurrence matrix, LBP and statistical features, in combination with an SVM classifier [90]. Their work also integrated a simple method for *tunnel* classification, a topic we will introduce in the next section, which derived the response of each of the images in the series to a Laplacian of Gaussian filter (LoG), and determined whether the sum of the resulting lumen or tunnel areas from the obtained images was greater than a predefined threshold in order to be classified as a tunnel frame. As the final output of their method in the detection of contractions, over CE videos obtained from ten different patients, their method achieves an average sensitivity of 70.08% with an average precision of 60.26%, indicating a significant proportion of false negatives and false positives respectively.

In this work, instead of explicitly detecting contraction events, we propose a method to automatically create an alternative for a manometric recording directly from the video resulting from a non-invasive CE procedure for computer-aided intestinal motility analysis. As the base of our method, we instead employ deep learning techniques to develop an accurate method for both tunnel detection and localisation based on simultaneous bounding box regression and classification using R-CNN-based methods. This will allow us to use the results not only use for the same purpose of aligning frames corresponding to the desired orientation of the capsule, namely those where the capsule faces the tunnel, but also to use the location and size of the bounding box for the extraction of features for the manometry approximation. We argue that it can be directly used to analyse a patient's intestinal motility much in

the same way as a manometric recording, while it can also serve as a helpful preprocessing procedure for any method to make use of intestinal motility.

Additionally, we argue that an accurate method of tunnel detection is useful on itself, providing an understanding of the pattern of changes in capsule orientation and movement throughout the CE procedure. In this way, it can also be useful for localisation purposes, as this information can be valuable for the deduction of the relative location of the capsule throughout the procedure. As a specific other use case that would benefit both from the tunnel detection method on itself and from the final result, we would like to highlight automatic capsule navigation. While there are still important mechanical difficulties to overcome for actively controlled capsules, automatic detection of the tunnel could help to correct the orientation of the capsule for stand-alone or assisting navigation algorithms. Direct analysis of intestinal motion, on the other hand, could prove to be helpful to develop mechanisms that could counteract intestinal motility for the capsule to stay on track. Although this is a side track of our method that interests us, we leave it out of scope for the remainder of this work to focus our efforts on intestinal motility visualisation.

4.2 Tunnel Detection

As our method is built upon tunnel detection, we first provide a formal definition thereof and discuss the surrounding issues we encountered. We define tunnel detection as detecting and localising the area in each frame that shows the anterograde pathway through the intestinal tract, hereafter generally referred to as the *tunnel* in either closed or open state. In earlier work on tunnel detection, we found this is often simplified to lumen detection in non-contracted state [101, 24]. The lumen is the inner space of the intestine, which theoretically always surrounds the capsule, but is only visible as a dark hole when the intestinal muscles are in a relaxed state. In contrast, in a contracted state the lumen is not directly visible. This can be observed in Figure 4.1, where the left-most image shows the small intestine in a fully contracted state in which we cannot actually observe the hole of the tunnel, while the right-most image is taken in the relaxed state and the tunnel is clearly visible as a dark hole. Namely, smooth circular muscles that are responsible for intestinal motility are located along the entire wall of the small intestine and frequently contract throughout the process of digestion, which occurs in the case of the endoscopic capsule just as it does with food. One important type of such motility is peristalsis, used in the small intestine to gradually transport the

(a) (b) (c)

Figure 4.1: Selected images from our data set, where we can observe that the "tunnel" in different states of intestinal muscle has notably different visual properties. Here (a) and (b) show the contracted and non-contracted states respectively, while (c) shows a semi-contracted state.

chyme to the colon, by repeatedly relaxing the circular muscles just in front of swallowed food, called chyme after the processing in the stomach, while contracting the muscles behind it. In fact, the current capsule endoscopes rely on this process to make their way through the intestine.

However, the predominant type of motility in the small intestine is segmentation contractions [30]. These contractions of the same intestinal muscles are not meant to transport the chyme further, but to mix the chyme with the secretions of the intestine, due to which they also often also push the food back, allowing greater mixing. Therefore, especially in the small intestine, naturally we frequently observe the state of contraction. In traditional endoscopy, techniques exist to infuse air into the stomach or intestine to open up the pathway and in fact may even be required to determine the direction of scope advancement [47], while no currently approved capsule endoscope disposes of an air infusion method. Therefore, in contrast to the previous work we explored, it is not only important to consider the visible intestinal lumen in the state of relaxation, but also the centre of the contracted muscles in state of contraction. We thus aimed to detect the direction of interest in all states of contraction as shown in Figure 4.1. For simplicity, we will refer to the point of advancement as the tunnel in all cases.

The recently emerged CNN object detectors are highly suitable for our tunnel detection purpose, as it has been shown that they can be complete, accurate object detectors, which are by design significantly faster than the original sliding window approach used in combination with complete CNNs to evaluate each sliding window. As we aim for our tunnel detection to be used as a preprocessing method, we limited ourselves to the sub-family of object detectors

that do not work with region proposals and have yet been shown to be highly capable of processing frames at a high frame rate on PCs. Even though we must account for this frame rate to drop significantly in the limited environment of the capsule endoscope, we only require it to run with a frame rate of 2 to 6 frames per second, as the cameras in popular current capsule endoscopes record at such a rate. Specifically, we chose for the You Only Look Once (YOLO) v3 method, as this is fully convolutional approach operates at a frame rate significantly higher than other approaches, while detecting objects of different sizes with acceptable accuracy on the COCO benchmarks. We hypothesize that in our limited case of having only one detection class, namely the tunnel in different states of contraction, as opposed to the common benchmarks, e.g. on the COCO data set with 80 different classes, we should be able to achieve an even more satisfactory result.

Note that the methods discussed above are a more resource-efficient and ultimately more powerful class of deep learning detection methods than the sliding window method integrated into our method in chapter 3, as we already briefly discussed there as well, since both the classification and localisation can be optimised simultaneously. In contrast with the problem faced there, the conditions of tunnel detection are favourable for their use, as here we are dealing with an object that does have better defined borders and shape. Additionally, for this objective we made a different choice in the trade-off between accuracy and resource efficiency due to the intention of the method to be used as a preprocessing method, as discussed above.

4.3 R-CNN Fundamentals

4.3.1 Introduction

Even though CNNs have proven their usefulness in object classification in combination with automatic feature learning on images, until recently it had its limitations when used for object detection. Object detection, in contrast to object classification, is not only recognising that an image contains an object of a certain class, but also where in the image this object is located. One way to do this using CNNs, which we have also done in our work here, is to divide an image into subregions and feed those subregions to the CNN instead of the entire image, i.e. the sliding window approach we introduced in Section 3.2.1. For this to work, the CNN needs to have been trained on comparable image regions instead of entire images. One downside is thus that they are difficult

to reuse when one wants to change region properties, such as the size for example, for which the network would usually have to be retrained on regions with the changed properties. Another downside of this method is that we need to process each region of the image through the entire CNN individually to finally obtain a class prediction for each of them, thus missing information that may depend on other regions in the image, such as the relative position. At the same time, processing a CNN once for the entire image instead of once for each region allows us to significantly improve the computational efficiency of the network.

Ideally, we would want to process the network only once for each image, thus increasing the efficiency through single matrix multiplications in the prediction phase while allowing us to take further advantage of this in backpropagation in the training phase. One way to achieve this in the prediction phase, is to adapt the CNN architecture from the sliding window approach from before. Namely, by converting the fully connected layers of the CNN to convolutional layers, we can obtain an output feature map instead of single class confidence values. If we then provide the whole image as input, each element of this feature map will correspond to a certain region of the image, which is equivalent to separately processing each patch obtained through the sliding window approach. However, as the size of the region depends on the architecture in combination with the size of the input images, this option limits us in our choices at the time of the network design to obtain the correct size of the output feature map and the desired receptive fields. Namely, in this case the size of the regions results to be equal to the factor by which the feature map was downscaled before the original fully connected layers, somewhat confusingly often referred to as the *network stride*, while the depth of the network and the size of the convolutional kernels together determine the size of the receptive fields. Additionally, this method also does not allow us to detect areas of arbitrary aspect ratios and different sizes, as all of the values in the output feature map then correspond to a region of fixed size with respect to the input size.

To solve these issues of using regular CNNs for the purpose of detection, the *Regions with CNN features* (R-CNN) method was suggested as an extension of the traditional object detection pipeline [27]. This method allows for obtaining more precise feature-based region proposals, instead of region extraction methods that ignore the content such as those based on regular sampling or the region sampling resulting from the CNN design in the aforementioned method. The idea behind the original method was to integrate a smart region proposal method into the overall detection method, classifying the proposed regions with a CNN. This original approach and later approaches that kept

this region proposal aspect are therefore often denoted *proposal-based*. As this approach was further built upon by researchers, later approaches were developed that no longer incorporated a separate region-proposal element and instead only employ a single connected network architecture, essentially regressing both bounding box coordinates and classification results in a single feature map. This class of approaches is denoted regression-based. Regression-based approaches have both advantages and disadvantages to proposal-based, however, and were therefore not widely adopted to replace them under all circumstances, with the result that both branches are being further developed in parallel. Recent work has also suggested to combine the best elements of both worlds [50].

The general approach of proposal-based methods is given in Figure 4.2, while the general approach of regression-based methods is given in Figure 4.3. Note that neither of these examples correspond to any architecture in particular. They are simply meant to indicate the main differences in the pipeline of both classes of methods. In both proposal-based methods and regression-based methods we can often distinguish two parts, namely the backbone and the head. The backbone is the base network that essentially serves as the feature extractor. The head is the network that classifies the features and regresses or refines bounding box coordinates. Although some methods introduce their own backbone, while other methods use popular CNN architectures such as VGG, we we can often interchange the backbone for other architectures. In doing so, we may have to pay attention to stride to avoid dimensionality issues and ensure that the architecture is valid, especially in those architectures that have additional shortcuts or branches to the head from within the backbone. The head is the part of the architecture that further regresses those features to predict class confidence scores and bounding box coordinates. Notably, R-CNN was the pioneer method in converting CNNs in full object detectors and therefore in this chapter we refer to all of these approaches as R-CNN-based. As these methods are not mere classification networks but basically full detection systems, we will also refer to them as detectors to distinguish them from classification CNNs.

Before we further extend on concrete R-CNN methods or architectures, there are some general concepts that are fundamental for these approaches, which we elaborate on here. Each of these concepts is either relevant to all R-CNN-based approaches or only to the proposal-based ones, but we consider them fundamental R-CNN-based techniques in general. Namely, knowledge of the techniques that are only commonly used in the proposal-based approaches can contribute to a better understanding for how the purpose of these concepts is

Figure 4.2: An example architecture of proposal-based approaches. Its main characteristic is the existence of a specific region proposal method, which is either directly connected to the input or to one of the (shared) convolutional layers of the overall architecture. This method proposes regions, over which features are then pooled from the convolutional feature map and processed further through a region-specific method.

Figure 4.3: An example architecture of regression-based approaches. Instead of employing an intermediate region proposal method, it both regresses bounding box coordinates and classification scores directly from the input image pixels through a single network architecture. It may however have different multiple regression branches that branch from intermediate convolutional layers of different sizes, with the goal to detect objects of multiple sizes on multiple scales.

achieved in alternative ways in regression-based approaches. Here we identify four important fundamentals that are substantially different from techniques in traditional CNNs: informed region of interest (RoI) proposals, RoI pooling, bounding box anchors and optimisation in R-CNNs.

4.3.2 Informed Region of Interest Proposals

The proposal-based methods heavily rely on the method employed to propose regions, and by inference on the features on which it bases its decision. As we shall see, the exact methods used differ for different variants of R-CNN, as many of those variants aimed exactly at improving the region proposal method or the manner in which it is incorporated in the overall detector. Here we distinguish between two types of methods, namely the deterministic methods, which find suitable regions based on predefined conditions or features, and the learning methods, which attempt to learn appropriate regions for the specific case. In essence, all of these methods are in a way generic object detectors, as their purpose is to return regions that correspond to a likely object.

Methods of the first type, the deterministic methods, can usually be interchanged for other classic region proposal or region growing methods. By far the most used method in literature is selective search, which is the method first suggested in the first original papers on R-CNN. We will therefore explain selective search in detail and give the reader an intuition of how this method may be replaced by methods that work similarly. Selective search is a bottom-up region growing or grouping approach. It defines different similarity measures between regions, in order to merge the most similar ones. In doing so, it inherits ideas from traditional image segmentation and exhaustive search, but instead of focusing on single features, the authors chose to diversify the search for similar regions to merge together through applying more diverse similarity measures [88]. Similarity is defined in a broad sense, as in fact it does not only measure similarity between two separate regions, but rather how well these regions fit together. It should be noted that these similarity measures in fact consider features that are not necessarily connected to the features we are looking for in the main R-CNN method. Concretely, the method uses predefined colour features, texture features, size features and location features to determine which two regions in the image are most similar. Additionally, they are chosen in such a way that merging the features of the most similar regions can be done through directly combining the overall features of both regions, without needing to recalculate the features from the pixel values for the merged region. This merging step is repeated until the whole image is

combined into a single region. Finally, the whole procedure is repeated for different colour spaces, and possibly other variants of the algorithm, thus yielding many different results that are ordered to allow the specific implementation of the method to make a trade-off between quality and quantity of found regions. In the original R-CNN, for example, the first 2000 samples of this ordered list were retained for further processing. The authors of the method suggest ordering this list in the order the regions were found, i.e. from last found to first found, although they introduce a degree of randomness in the sorting algorithm to prevent the method from too heavily emphasising the large regions. They do so by multiplying the rank of each of the image regions in the sorted list by a random value, and finally sorting the entire list of objects by the thus obtained values. When using bounding boxes and wanting to remove duplicates, they note it is favourable to first sort the entire list in the explained manner, and afterwards remove only the lower-ranked duplicates. This also favours bounding boxes with duplicate detections, which the authors argue to be desirable as objects that are proposed in multiple grouping strategies are more likely to be visually coherent and thus to represent an object.

The second type of method, the learning methods, are commonly CNN architectures that are integrated into the overall architecture and are referred to as Region Proposal Networks (RPNs). These architectures take an entire image as input and produce an output feature map where each element contains the probability that said element is part of an object, as well as the parametrised coordinates of the corresponding bounding boxes, thus generating an output of $h \times w \times k$, with h and w being the input image height and width respectively and k being the number of bounding boxes we allow to be detected at each element of the feature map. It is important to note that k has to be set in advance and is thus a hyperparameter of the CNN architecture, as the method regresses fixed size feature maps with bounding box coordinates and classification of region proposals in a single feature map directly from the entire image input. The bounding box coordinates are in fact not any absolute coordinates, but are predicted relative to each of the k *anchor boxes* that have a predefined width and height, with each anchor box corresponding to one of the output bounding boxes. We will explain this important concept in more detail in Section 4.3.4, along with the motivation of the necessity for them.

An important aspect of the RPN is its loss function and how it deals with class balancing between positive samples and negative samples. Using an anchor box approach, the number of negative samples commonly greatly outweighs the number of positive samples, as will become clear when we discuss anchor boxes in detail later on. This issue was already targeted by the authors of

the first R-CNN-based method that implemented RPNs [72], who suggested to take this into account by applying undersampling of the negative class in the loss function. Concretely, they aim for a 1:1 ratio between positive and negative samples by randomly sampling a fixed-size *mini-batch* (256 samples in their case) of region proposals in the loss calculation, which in their case should thus contain 128 samples of each class. Each time an image contains too few positive samples to meet this ratio, the remainder of the batch is still padded with negative samples to complete the mini batch. In following R-CNN-based approaches, the approach of balancing positive and negative classes in the loss function was further extended on, e.g. by changing the ratio (e.g. 1:3 [32]) or by using weighted loss functions.

Since the RPN outputs the classification probability as well as the regressed bounding box coordinates, both components participate in the loss function of the network. The loss function essentially defines a sum of the classification loss and the regression loss, both of which are normalised with respect to the number of bounding boxes involved, weighted by a parameter λ that controls the trade-off between the two. This is similar to the loss function of regression-based approaches, as both approaches in fact rely on similar principles in general. In fact, RPNs are regression-based detectors that detect generic objects instead of multiple classified objects, thus producing an output feature map that contains objectness scores along with the region coordinates, whereas full regression-based detectors output classification scores instead of objectness. Therefore, we will leave the details with respect to the loss function of the RPN to Section 4.3.6, where we discuss the different loss functions in R-CNNs in general. In some approaches the RPN is trained separately from the rest of the detector architecture, while in other approaches it is integrated with the overall architecture and trained simultaneously while sharing features, commonly alternating between freezing the weights of the RPN and freezing the weights of main network to obtain appropriate weights for the shared convolutional layers.

4.3.3 RoI Pooling

After obtaining regions using any of the region proposal methods, they need to be further processed through the remainder of the R-CNN architecture to predict classification results and final bounding boxes for each of the regions. In R-CNN-based methods, it is interesting to first process the entire image through several (convolutional) layers before we continue on a per-region basis for efficiency purposes, as we then need to extract overall base features only

once for all regions, thus saving redundant per-region operations. However, in doing so, it is no longer straight-forward which of the features in the feature map resulting from these shared layers correspond to each of the regions found by the region proposal method. Additionally, the region-specific part of the architecture requires a fixed input size that should be equal for all regions, for which the features of each of these regions need to be mapped to a feature map of the required size, regardless of their different original sizes. The RoI pooling layer is a special type of layer that can perform both of these operations in a single layer and output a feature map of a predetermined fixed size, thus serving as a bridge between the base architecture and the region-specific part of the architecture.

RoI pooling derives its name from the fact that it performs a pooling operation comparable to the operation in the common pooling layers in CNNs described in Section A.4.2. However, here the goal is to pool exactly those features corresponding to the RoI from the feature map derived from the entire image in such a way that the layer can be shared among RoIs and the output size is constant and thus does not depend on the size of the RoIs [26], as this should be equal for all of the regions for the detection network to ensure a valid architecture. Therefore, for greater regions, these features need to be pulled over a more extended area. This is where it differs from normal pooling layers, for which the pooling operation has a fixed receptive field (commonly 2×2 or 3×3), while the stride of the resulting feature map is a dependent variable that we have to take into account when ensuring the validity of the architecture we define. In RoI pooling, we instead have a fixed output size, while the resulting stride and receptive field, or the pooling window size, are dependent variables. For example, if we require an output feature map of $h_o \times w_o$ cells, we need to use a pooling window of $(\frac{h_i}{h_o} \times \frac{w_i}{w_o})$ for that specific input region or sample.

The best way to understand these concepts is by a concrete visual example, as shown in Figure 4.4. In this simplified example, we have an output feature map from a shared convolutional layer of size 7×7 and a desired input size of 2×2 for the region-specific part of the architecture. In our example, we also have coordinates of two different RoIs as a result from our proposal method, each of which RoIs is provided as input to the RoI pooling layer. Let us suppose that those regions correspond exactly to the dashed lines visualised in Figure 4.4. As the coordinates of the RoIs are defined at a higher precision than our feature map at this point, the position values corresponding to the found RoIs are decimal values in a continuous range, while the feature map values are only defined on a discrete range. The original RoI pooling method

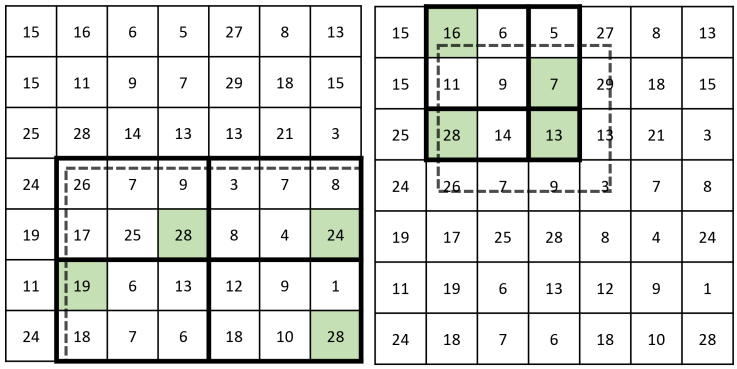


Figure 4.4: RoI pooling visualised for a 2D convolutional feature map of 7×7 and two proposed regions from the region proposal method. Figure (a) shows the case for the first region of size 6×4 on the convolutional feature map, while Figure (b) shows the case for the second region with odd dimensions of size 3×3, where the pooling sections differ in size. The green backgrounds indicate the maximum values in each of the pooling sections, which will thus form the output of the RoI pooling layer.

dealt with this by first quantising the coordinates are then first quantised to include all of the cells they overlap. This is commonly done by converting the coordinates at the edges of the RoI from (x, y) to $(\lfloor x \rfloor, \lfloor y \rfloor)$, as this is efficiently implemented as an integer cast.

In the case of the first region, the RoI located in the right-bottom corner of the image as indicated in Figure 4.4a, would then be of size 6×4. In this case, each pooling section will be of size $\frac{6}{2} \times \frac{4}{2}$, i.e. 3×2. The second RoI is of size 3×3 on our feature map and is located close to the upper left corner as shown in Figure 4.4b. As the resulting dimensions of the latter RoI are odd, the same division for this RoI does does not yield integer results, due to which the different cells of the pooling window will not all have equal receptive field sizes. In this particular case, the four pooling sections will be of sizes 2×2, 2×1, 1×2 and 1×1 respectively.

As this example shows, RoI pooling will not always be precise. Namely, due to the quantisation of the coordinates, we may be losing information around certain edges of the RoI, while on other edges we may actually be pooling features that do not correspond to it. For example, in case of our 3×3 RoI, the coordinates of the top-left corner are $(1.25, 0.75)$. As this only partially covers the feature map cell at position $(1, 0)$ while we are pooling all the information

from it, we are in fact pooling more information than really should be the case. At the right and bottom sides of our RoI, the opposite happens, where the cells at the fourth column and fourth row respectively are not taken into account, even though our RoI covers a substantial part of them. Additionally, we cannot guarantee the dimensions of the input region to be a multiple of $h_o \times w_o$, due to which the pooling windows are not of equal size all over the input, and they will be sized differently near one of the edges of the image than near the other in that case, e.g. with a subset of the pooling windows having dimensions $\lfloor \frac{h_i}{h_o} \rfloor \times \lfloor \frac{w_i}{w_o} \rfloor$, while the remaining windows would differ in at least one of the dimensions, i.e. $w_p = w_i - (w_o - 1)\lceil \frac{w_i}{w_o} \rceil$ or $h_p = h_i - (h_o - 1)\lceil \frac{h_i}{h_o} \rceil$, depending on the exact implementation.

In order to deal with these issues, improved methods have been suggested that make use of bilinear interpolation on the feature map. The two most popular methods of these are RoI Warp and RoI Align. RoI Warp deals with the problem of entirely losing information around the edges by introducing bilinear interpolation within the pooling windows. In this manner, instead of having the pooling windows we found before, we can define four pooling windows of equal size, namely 1.5×1.5 each, while we regularly sample n data points in each of the pooling windows at positions $(\frac{i}{n+1}, \frac{j}{n+1})$, for each combination of $i = 1 \dots n$ and $j = 1 \dots n$. RoI Align takes this idea one step further by dropping quantisation altogether and aligning the sampled points purely to the RoI instead, effectively aligning the data points to the RoI as its name suggests.

4.3.4 Bounding Box Anchors

One way to obtain bounding boxes from input images through an R-CNN-based approach would be to directly predict bounding box properties for a detected object, i.e. location and dimensions. However, this approach is prone to errors because it tends to favour bounding boxes with large dimensions, while at the same time it destabilises the training process, since the range of values to predict varies significantly between samples, while it leads to further issues in the case of predicting multiple objects of different aspect ratios. Therefore, an often used technique in the methods elaborated on in this section is to predict such coordinates as translations and scaling factors of a set of boxes with predefined aspect ratios, also called *anchor boxes*, which makes it easier for a network to learn [69]. As in the R-CNN architecture every cell of the output feature map independently makes predictions with respect to each of the defined anchor boxes, we can essentially view these

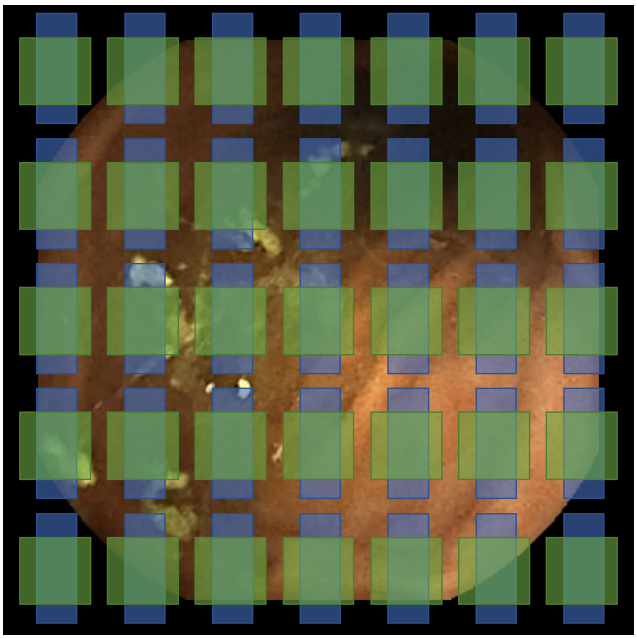


Figure 4.5: An example of two anchor boxes of different aspect ratios, tiled across a CE image for exemplary purposes. The aspect ratios of the anchor boxes heavily depends on the initialisation, although they can be refined through training. The number of anchor boxes depends entirely on the network configuration, while whether they overlap or not and to what extent largely does as well.

boxes as being tiled across the image as visualised in Figure 4.5. In that example we used two anchor boxes with a different aspect ratio. The predefined anchor boxes can thus be seen as a static part of the network architecture, whereas the cells essentially predict an instance of each anchor box with specific parameters, notably the specific translation and scaling factor, along with the class confidence scores. The network learns to predict these parameters from the ground truth bounding box, using bounding box overlap metrics such as intersection over union (IoU) in the loss function of the network during the training phase with the purpose to increase IoU values with the ground truth.

From the above, we can understand that anchor box definitions have a vital role in the detection performance of the R-CNN. Poorly defined anchor boxes, e.g. with aspect ratios that do not correspond to our domain, negatively affect performance. It is therefore important that the used anchor boxes are relevant to the specific detection problem. It can be observed that in many

cases, objects of a specific class of our problem have characteristic aspect ratios, such as pedestrians, cars, for example. By defining our anchor boxes in accordance with those aspect ratios, predictors (i.e. a set of output values forming a single bounding box prediction) linked to a certain anchor box can more easily become specialised in the detection of the class of objects that have a characteristic shape with an aspect ratio similar to that of the anchor box. To generate boxes that are relevant for the specific data or the domain of the problem at hand, different methods have been suggested, of which the most popular appears to be the use of clustering methods over box sizes or aspect ratios on the training dimensions, as first proposed in YOLOv2 [69].

In most implementations, there is a fixed number of predictors to output a fixed number of multiple bounding boxes per image or per grid cell. In this case, each bounding box predictor corresponds to one anchor box and the predictions are somehow relative to the anchor box properties, such as positional offsets and scaling factors. For example, in the case of YOLOv3, as we shall see, the output that relates to the positional offset is first passed through a sigmoid function and then summed to the coordinates of the grid cell, while the output values that relate to the dimensions are passed through the natural exponential function, then multiplied by the corresponding bounding box anchor dimensions, and subsequently multiplied by the image size, as the values were normalised.

4.3.5 Bounding Box Regression

Before we delve deeper into the other concepts of R-CNN, it is important to understand the concept behind bounding box regression, that all methods discussed here employ in one way or another. Although they may all differ in the exact implementation of this regression and how this is combined with the classification of the regions, bounding box regression generally refers to the regression of the parameters that determine the extent of the bounding boxes, which commonly relate to their x- and y-positions, their width and their height, thus yielding four coordinates. Even though in the proposal-based methods we already obtain regions as the input for the classification part of R-CNN head, in many proposal-based methods bounding boxes are still regressed by means of refining the first proposed region, to fit the boxes more tightly around the objects of a certain class contained within. In regression-based approaches, bounding box coordinates are instead regressed directly alongside the classification scores. In all cases, the regression targets of the coordinates are usually defined relative to other boxes, whether it be anchor boxes or the

boxes found in the region proposal method, which often leads to improved stability as explained in Section 4.3.4, with regression thus working similarly in all cases.

In this section we explain the regression targets of the bounding box coordinates by the example of the original R-CNN approach with region proposals from the selective search algorithm, while we also largely following the explanation in the original paper [27]. In this case, we thus already deal with proposed regions from proposal-based methods as input of the network head, instead of entire images. Also, note that here we no longer operate on the original image input, but instead on the shared feature map that is the output of the backbone, as addressed in Section 4.3.3.

Essentially, in training an R-CNN detector with region proposals, the goal is to map the bounding box coordinates obtained from the region proposal method to a corrected bounding box that fits better to the ground truth box. In other words, we want to find a function that maps the proposal box coordinates (p_x, p_y, p_w, p_h), hereafter denoted $\mathbf{p}$, to the ground truth box coordinates (g_x, g_y, g_w, g_h), hereafter denoted $\mathbf{g}$. The parameters we want to estimate are not the direct coordinates, however, as these can be within a wide range of numbers. Instead, the authors of R-CNN suggest to parametrise them using scale-invariant translations for the location coordinates p_x and p_y, while using log-space translations of the width and the height [27]. The values we actually want to regress are then defined as d_*, where $d_* = w_*^T \phi_k$, with k being the index of the last feature map layer of the network, i.e. a weighted combination of the features returned by the CNN. The true estimated coordinates $\hat{\mathbf{g}}$ of the ground truth boxes can then be defined in terms of transformations on these functions through the following transformations on d_*:

$$\begin{aligned}
\hat{g_x} &= p_w d_x(\phi(\mathbf{p})) + p_x, \\
\hat{g_y} &= p_h d_y(\phi(\mathbf{p})) + p_y, \\
\hat{g_w} &= p_w \exp(d_w(\mathbf{p})), \\
\hat{g_h} &= p_h \exp(d_h(\mathbf{p})).
\end{aligned}$$

As the functions d_* need to be optimised for, the authors model the ground truth boxes analogously in terms of the ground truth boxes $\mathbf{g}$, such that we

obtain the regression targets t_*:

$$
\begin{aligned}
t_x &= \frac{g_x - p_x}{p_w}, \\
t_y &= \frac{g_y - p_y}{p_h}, \\
t_w &= \frac{\log g_w}{p_w}, \\
t_h &= \frac{\log g_h}{p_h}.
\end{aligned}
$$

Using these regression targets, the problem can be modelled as an ordinary least squares regression problem:

$$
w_* = \underset{\hat{w}_*}{\operatorname{argmin}} \sum_{i}^{N} \left(t_*^i - \hat{w}_*^T \phi_k(\mathbf{p}^{(\mathbf{i})}) \right)^2 + \lambda ||\hat{w}_*||^2.
$$

In the Fast R-CNN method and later methods, the bounding box parameters are instead regressed through a connecting MLP with one or more layers [72]. In the case of a layer with a linear activation function this is equivalent, while in the case of non-linear activations it can define more complex relationships. It should be noted that this comes at the cost of efficiency, considering the much less efficient stochastic optimisation of neural networks.

4.3.6 Loss Functions

In contrast to classification networks, in the case of R-CNN approaches both class confidence scores and bounding box parameters are regressed simultaneously. Therefore, additionally to the classification error, the loss functions should take the localisation error into account. As a consequence, these often become a sum of both components, weighted with a regulating factor λ, i.e.

$$
L(\hat{p}, p, \hat{t}, t) = L_{\text{cls}}(\hat{p}, p) + \lambda \mathbf{1}_{p \geq 1} L_{\text{loc}}(\hat{t}, t),
$$

where $\hat{p}$ and $\hat{t}$ correspond to the predicted class confidence scores and bounding box regression values respectively, while p and t correspond to the true classes and bounding box regression targets respectively. For bounding box regression, smooth L_1-loss is a common choice as well as L_2-loss, while for classification cross-entropy is commonly chosen as we have seen for classification with CNNs before. Some methods use hinge-loss for classification instead, as was the case for the first R-CNN architecture, which used SVM instead of an MLP. The

details of all these specific loss functions are discussed in the appendix on CNNs in Section A.5.

In case of integrated region proposals, the region proposal loss is regressed simultaneously. This often occurs in the case of a connected RPN or fully regression-based approaches, which thus incorporate an additional component in the loss function that corresponds to the quality of the region proposals. This component is calculated similarly to the classification error, but this time simply based on the two classes "object" and "no object". In methods that are based on a separately trained RPN, the loss for this regression is a different loss function altogether, commonly only including the previously mentioned objectness classification and the bounding box regression. As the proposals are not connected to the final bounding box and class predictions of the overall detector, it is often desirable in this case to have a particularly high recall, as boxes that are wrongfully not detected in this stage, can never be detected in the second stage.

4.3.7 Model Evaluation

While we already discussed evaluation of CNNs used for classification in Section 3.2.3, the networks we discuss in this chapter are full object detectors that predict bounding boxes on top of classification. Therefore, we require different evaluation metrics. For object detectors in general, a specific evaluation metric that is widely used is the average precision (AP). This measure was first introduced by the popular PASCAL VOC object detection challenges, after first making its introduction in the challenge of 2007 [22], and has since then been widely adopted as evaluation metric in object detection papers and in other challenges, such as COCO. When using all data points, its value is calculated per class by first collecting all the individual detections for that class over all images and sorting them based on their confidence scores in descending order, and at each rank in that list summing the precision values multiplied with the difference in recall value compared to the last rank, which is equivalent to a setting the threshold at that confidence score. The formula for AP is given by

$$AP = \sum_{n}^{N} (R_n - R_{n-1}) P_n, \tag{4.1}$$

where R_0 is 0. This is exactly the area under the precision-recall curve obtained from plotting the recall and precision values at each rank in the list against each other. In the earlier challenges, the organization of PASCAL VOC used an

interpolated AP which sampled the AP at 11 fixed, uniformly spaced intervals, attempting to even out the small "wiggles" usually present in the precision-recall curve caused by small variations in the ranking of the samples. From 2010 onwards, they decided to start using Eq. 4.1 as they concluded the interpolated AP was not discriminative enough at low AP [21].

As can be understood from the above, the AP is very sensitive to the relative ranking of the detected objects and severely punishes false object detections in the high ranks, as well as a low detection rate, i.e. low precision values at low recall values. Reviewing both of these measures together while not directly considering the negative class is necessary especially in case of an imbalance towards negatives, which is practically always the case in object detection as background and other objects occupy significantly more area in the input images than the object itself. By considering the entire relative ranking, i.e. the entire area below precision-recall curve, we obtain information about the robustness of a model in the sense that a good model should be able to return relatively many true positive detections before returning false positives. In our case, we also decide for using AP as an evaluation metric as calculated in Eq. 4.1, since the objects we aim to detect also occupy significantly less area than the background. Furthermore, we consider that this metric depends on all terms of the confusion matrix except for the true negatives, which is exactly what we are least interested in.

4.4 Proposal-based Detectors

The original R-CNN method consists in using a sophisticated method to generate about 2000 class-independent region proposals for each input image at test time, from which it then extracts and classifies feature vectors. Even though the authors chose for the selective search algorithm to generate region proposals, R-CNN can be used with any chosen region proposal method. The images corresponding to each region proposal are then resized (and possibly warped) to the input size required by the employed CNN architecture in order to obtain a fixed-length feature vector for each region from the CNN. Finally, the obtained feature vectors are classified through class-specific linear SVM models. The authors also suggested performing further bounding box regression to reduce localization errors introduced by the region proposal method, which they did through a linear regression model to predict a new detection window given the deepest spatial features of the CNN.

Fast R-CNN [26] introduced innovations to this method that mainly allowed for faster training and prediction times. Namely, instead of processing the whole CNN for each region, the authors proposed to first process the whole image through the CNN and then extract the features per region proposal, or region of interest (RoI). This idea was based on modifications to R-CNN by SPPnet, which already proposed to speed up R-CNN by sharing computation through the introduction of a shared convolutional feature map which it extracts the feature vectors from for each of the proposals, using spatial pyramid pooling. However, SPPnet is still a multi-stage pipeline that involves feature extraction, a fine-tuning stage in which earlier feature extraction layers are fixed, training SVMs and fitting bounding-box regressors, while, in contrast to R-CNN, it does not allow for updates to the convolutional layers preceding the spatial pyramid pooling during fine-tuning. Fast R-CNN instead has a single-stage training procedure, which can update all network layers. Features are pooled from the convolutional feature map into to a fixed-size feature map with a width of W and a height of H through RoI pooling, which we discussed in Section 4.3.3. These W and H parameters are incorporated into the CNN architecture and therefore hyperparameters of the model. While we already discussed how RoI pooling operates to output a feature map of a constant size, we can now see how configuring the size of the output feature map inversely determines the extent of the pooling operation. Namely, if the width and height of the region corresponding to a RoI are w and h respectively on the convolutional feature map derived from the entire image, the size of each pooling section would be approximately $\frac{w}{W} \times \frac{h}{H}$, but would be subject to quantisation as we only deal with integer values. Instead of using separate SVMs for classification, Fast R-CNN uses a fully connected layer with a softmax activation function on the feature vector on one hand to obtain the class probabilities, while on the other hand they use a parallel fully connected layer on the feature vector to perform the bounding box regression.

Finally, building forth on Fast R-CNN, Faster R-CNN was developed. The main innovation of this method was a significant reduction of the number of region proposals, while ensuring a more accurate object detection. Namely, in Fast R-CNN the region proposal method was separate from the remainder of the detector, due to which the method needed to have a specifically high recall often at the cost of a lower sensitivity, as briefly explained at the end of Section 4.3.6. Faster R-CNN instead introduced region proposals through an integrated RPN as explained in Section 4.3.2 which operates on the feature map generated by layers shared with the backbone CNN. The region proposal model outputs both a bounding box prediction and a binary class prediction, which is either object or background, after which non-maximum suppression

is used to eliminate similar bounding boxes that are predicted to correspond to an object. This region proposal network is trained together with the rest of the model and can finally learn to generate high-quality and precise region proposals, which significantly reduces the number of proposed regions. As in Fast R-CNN, RoI pooling is then applied to the output of the first CNN for each proposed region after which through several fully connected layers finally a class is predicted along with a bounding box. Mask R-CNN takes the idea of Fast R-CNN a step further by replacing the RoI pooling layer by a RoI align layer and adding a fully convolutional network to the output in order to predict a pixel-level mask for each object. The RoI align layer removes the quantization applied in the RoI pooling layer due to sampling the RoI features only at discrete coordinates of the RoI projection on the feature map. This may not lead to wrong classification results as classification is robust to small translations, but it does cause a misalignment in the case of predicting pixel-accurate masks [32]. The RoI align layer instead applies bilinear interpolation to compute values at continuous locations, which has the additional advantage of being able to set the number of data points to sample from for each value of the output feature map.

4.5 Regression-based Detectors

The other type of CNN object detectors are the regression-based, which first emerged from the need to improve the speed for implementation on devices with fewer resources. In these approaches, regression-based refers to the fact that bounding box coordinates are directly regressed from the input image pixels into the output feature map in a single stage, thus entirely eliminating the expensive step of intermediate region proposals. Despite the name, they also still perform classification simultaneously, combining both aspects into a single loss function during training. Pioneer methods among these were MultiBox and AttentionNet, which did not yet perform well enough to compete with the region proposal-based methods.

Later approaches further refined the ideas presented in those methods, starting with the You Only Look Once (YOLO) method [68], which was later redesigned to improve detection rates of smaller objects specifically through YOLOv2 [69] and YOLOv3 [70]. The original version of YOLO introduced the concept that the central box in the last feature map of the network was responsible for predicting the presence of an object along with its bounding box coordinates and classification. In YOLOv2 the concept of anchor boxes was adopted to

control the gradients in the bounding box regression, which was found to also improve detection rates if the dimensions of the anchor boxes were chosen wisely, basically making anchor boxes specialists in detecting certain objects with a typical aspect ratio. As the authors of YOLO continued to improve their method, YOLOv3 still provides a highly competitive accuracy as compared to other R-CNN-based approaches, especially considering its inference times. In between iterations of YOLO, other competitive methods were suggested that dealt with issues in earlier YOLO versions.

One of these in particular became a popular approach by itself, namely Single Shot Detector (SSD) [53], which also seems to have served as an inspiration of the YOLOv3 method. Namely, the older YOLO methods had difficulties in detecting small objects. SSD therefore used the observation that CNNs usually gradually reduce the size of the feature maps in the hidden layers to reduce dimensionality towards the final classification layers that eventually predict the classes. To be able to detect the smaller objects in a regression-based approach without the need for a region proposal method, they suggested an approach where bounding box regression was performed on intermediate layers of a CNN detector each time before pooling layers, resembling a classical feature pyramid approach, instead of only doing this for the smallest output layer. This approach also made use of anchor boxes that MultiBox also used, while the original method of YOLO did not. Additionally, this method made use of hard negative mining to address the extreme majority of negative samples as opposed to positive samples in object detectors, which was first used in Fast R-CNN. Using the observation that many of those samples are easily classified as background, hard negative mining limits the number of negatives included in the calculation of the loss function to an approximate 1:3 ratio between positive and negative samples after ranking the negatives by their loss value and including only those with the highest ones. SSD outperformed Faster R-CNN on the COCO benchmark [52] as well as YOLO and YOLOv2.

Apparently inspired by the SSD approach, several of its concepts were integrated in the method of YOLOv3. Namely, it included both the idea of regressing bounding box coordinates from output feature maps from intermediate layers of the CNN and the usage of anchor boxes, which were already adopted in YOLOv2 and further improved in YOLOv3. The authors also experimented with a recent innovation for multi-class object detection through regression-based models that was introduced in RetinaNet, namely the concept of *focal loss*. This concept extends the earlier mentioned hard negative mining defined in SSD and addresses the imbalance between positive and negative samples dynamically [51]. Concretely, it is a version of cross-entropy

loss extended by a scaling factor that decays to 0 as the confidence of correct classification decreases, thus forcing the model to focus on the hard samples. In this manner, the threshold on the inclusion of samples in the loss function is effectively continuously adjusted in the form of sample weights. However, the authors of YOLOv3 observed that this concept actually dropped average precision values in their method [70]. Nonetheless, the method of RetinaNet on itself is interesting as it achieves high average precision values on COCO, but at the cost of a significantly increased inference time. Another interesting recent method that obtained a higher average precision on the COCO data set than YOLOv3 is CornerNet [45], which introduced bounding box detection through detecting only its top-left and bottom-right corners, eliminating the need for anchor boxes altogether. They also introduced corner pooling to train the network to better localize the corners.

With the shift from classification to full object detection using bounding boxes, the loss functions of the CNNs also changed to incorporate localisation loss and objectness loss. While R-CNN still used SVM for classification of the regions, in Fast R-CNN this was done in the same network using a single loss function, which consisted of the weighted sum of the log-loss (or cross-entropy) for classification and a smooth L_1-loss over the bounding box coordinates for bounding box regression. Even though Faster R-CNN introduced the concept of anchor boxes, which was also adopted by the later approaches with fully convolutional elements, the loss functions always remained similar, using a weighted sum of cross-entropy for classification and a smooth L_1-loss for bounding box regression, while e.g. Mask R-CNN only introduced an additional loss for the segmentation mask. The original YOLO method, however, introduced a slightly more complicated loss function, as YOLO strictly determined that the cell of the feature map that corresponded to the centre of an object was responsible for the prediction of the corresponding bounding box and class scores. Therefore, the loss function only sums the loss over bounding box predictions and classification from those grid cells, while it does consider objectness scores from all cells. Namely, if the objectness score does not meet a predefined threshold, all other predicted values are ignored. Instead of using smooth L_1-loss, in YOLO all of these partial losses were defined as L_2-loss. It also had parameters to regularize the contribution of the localization error as well as the contribution of cells that had to predict no object. In later versions of YOLO, i.e. YOLOv2 and YOLOv3, the method changed considerably and with it did the loss function, as it was gradually transformed to include the concept of anchor boxes as well as new thresholds and regularization methods. This discussed in detail in the following section.

4.6 You Only Look Once (v3)

The method of YOLOv3 is one of the most recent popular variants on R-CNN focused on resource efficiency. It directly combines feature extraction, region proposals, final bounding box predictors and classification into a single fully convolutional network (FCN). It is thus designed to deduce class probabilities and bounding box coordinates simultaneously straight from the input pixels. Due to this, YOLOv3 requires a significantly lower processing time and computer resources or energy than most other R-CNN methods. An additional advantage of the fully convolutional manner is that it allows the method to implicitly encode contextual information, while the region proposal techniques or regular sampling used in most R-CNN methods do not consider the context of a RoI and can thus miss information that only becomes obvious by seeing the wider picture.

In the original version of YOLO [68], bounding boxes were predicted only from the latest, fully reduced feature maps, which resulted in smaller objects often being missed as the cells of those maps have a too large receptive area for small objects to play a significant role in the values at that stage. In YOLOv3 [70], bounding box predictions are made based on three distinct feature maps of three different sizes. According to its authors, this resulted in a higher detection rate of smaller objects, although detection of big and medium-sized objects was slightly sacrificed for it. We opted for this latest version as it seemed to offer an acceptable trade-off for us, hypothesizing that the case of a clearly visible, open tunnel is the most straight-forward case for detection, while the small centres of the contracted muscles are the most difficult to detect.

Despite consisting of a single FCN architecture, we can still distinguish two sub-networks in YOLOv3. Namely, the first of those sub-networks continuously decreases the feature map in size through convolutional layers with different strides and serves as main feature extractor, which we hereafter refer to as the base CNN, while the second sub-network, hereafter referred to as the top CNN, makes predictions using a concept similar to feature pyramid networks, namely by upscaling this feature map through several layers and merging the increased feature maps with feature maps of the same size from the base CNN by concatenation, hereafter referred to as the top CNN. For the base CNN the authors of YOLOv3 suggest their self-designed CNN architecture Darknet53, which we also used in our work, as they found it to obtain similar results to ResNet-152 with a significantly lower number of operations. Through its convolutional layers, this architecture downscales the input image

by a factor of 32, which is also referred to as a stride of 32 considering moving one cell in the final feature maps is equal to a distance of 32 pixels in the input image. For the top CNN, the authors propose a network that effectively makes predictions at three levels of a spatial pyramid. Namely, they first append several convolutional layers to the base CNN. The last of these outputs a 3-D tensor that simultaneously predicts bounding box coordinates, object probability and class probabilities [70]. This corresponds to three different bounding box predictions with corresponding probabilities per cell of the feature map. Separately, they also sum the feature map obtained from two layers before this first predicting layer to the last feature map from the base CNN of the same size, to get more meaningful and finer-grained semantic information. They then process this feature map through several more convolutional layers, of which the last one again makes predictions analogously to the previous predicting layer, although now twice the size. Analogously, the third and last predicting layer is again based on a stack of convolutional layers on top of the feature map from two layers back. This is visualised in Figure 4.6.

4.7 Experiments and Results

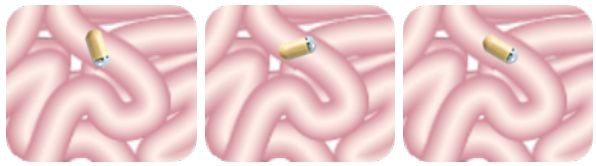
4.7.1 Data Preparation

4.7.1.1 Data Conditioning

We obtained our data in collaboration with Hospital Universitari i Politecnic La Fe, Spain. With the explicit consent from 10 different patients, we collected the videos of their capsule endoscopy procedures with the PillCam SB 3 capsule. Through the official software that ships with the capsule, we exported segments of those videos that correspond to the small intestine to a public format for further processing. From these videos, we then exported individual frames at a regular interval and eventually exported 1385 frames. Additionally, we collected five different videos that we use to evaluate our manometry approximation method.

The frames we obtained show different scenes from the intestine, which could be roughly divided into two different cases of interest: facing the wall (FW) and facing the tunnel (FT). This is visualised in Figure 4.7. Through a tool we developed for bounding box annotation, two expert readers observed all collected frames and carefully annotated the center of contraction and the lumen with bounding boxes. In some situations, even though we may be facing

Figure 4.6: An overview of the YOLOv3 model that we used for our detections. The dotted black lines represent the ResNet blocks of the Darknet53 base network that are not visualised for simplicity, while the top network shows all the convolutional layers, both the ones with a kernel size of 1 and those with a kernel size of 3. The straight thin blue arrows, connecting layers from the base network with the top network, represent concatenations. Detections are made at three different feature map sizes.



(a) (b) (c)

Figure 4.7: The two situations we aim to distinguish. Facing the wall (FW) in (a) and (b), and facing the tunnel (FT) in (c).

the center of contraction or the lumen in an open tunnel, we cannot observe it due to intestinal content occluding them. As it is relatively common to observe some intestinal content within the tunnel in non-contracted state, we still used those frames both in training and in testing without bounding box annotation to train our algorithm not to mistake intestinal content for the tunnel.

To deal with the black frame around our images, as already discussed in Chapter 3, we made use of the same mask as there, visualised in Figure 3.2b. In this case, as we use the entire image, we first replaced all the pixels corresponding to the black area of the mask by pure black pixels, to ensure any remainders of metadata usually present in the frame was removed. We then tightly cropped the image around the area of interest, including a minimal number of pixels from the black frame, to reduce dimensionality as much as possible without removing any useful information. In this way, from the image originally extracted at full resolution from the manufacturer's software of 576×576 pixels, we finally obtained images of 512×512 pixels, as shown in Figure 4.1, to serve as the input to our method.

4.7.1.2 Data Partitioning

For the learning procedure, we subsequently partitioned our data so that in our test set we would not have any images from the same patients whose images were used during the training procedure, in order to provide a fair evaluation of our method by testing that our method generalised not only to new images, but also to new patients. Therefore, we wrote a script that first randomly shuffled all the patients. Then, for each patient, we randomly removed frames from our data set until we had at most 5 positive frames and 5 negative frames of each patient. Subsequently, we created our data set by repeatedly reading the first patient from the list, adding the corresponding frames to the test set and removing that patient from the list, until we obtained the desired percentage

of frames in our test set, which we set to be 10%. For the remainder of the patients, we simply used a random partitioning by a hold-out of 20% to obtain our train and validation sets. In this way, we finally ended up with a training set of 921 frames, a validation set of 263 frames and a test set of 201 frames.

4.7.2 Tunnel Detection

In our tunnel detection experiment, our aim was to detect the tunnel both in contracted and in non-contracted state. For this purpose we used YOLOv3 as visualised in Figure 4.6. However, in the paper published about YOLOv3 [70] it remains unclear to what extent the author changed the loss function with regards to previous versions of YOLO. Namely, through the two subsequent iterations (YOLOv2 and YOLOv3), the method changed significantly, and the loss function must have changed along with it, as can also be deduced from the code in the official YOLO repository [67]. Therefore, we compared the loss function in the code we used against the loss function we deduced from the official repository, and found them to be equivalent. This loss function can be defined as follows:

$$
\begin{aligned}
L \quad = \quad & \sum_{i=0}^{B} f(w_i, y_i) H([x_i, y_i], [\hat{x_i}, \hat{y_i}]) \\
& + \sum_{j=0}^{B} \frac{1}{2} f(w_i, y_i) [(w_i - \hat{w}_i)^2 + (h_i - \hat{h}_i)^2] \\
& + \sum_{i=0}^{S^2} \sum_{j=0}^{\hat{B}} \mathbf{1}_{ij}^{obj} H(C_{ij}, \hat{C}_{ij}) \\
& + \sum_{i=0}^{S^2} \sum_{j=0}^{\hat{B}} \mathbf{1}_{ij}^{noobj} \mathbf{1}_{ij}^{IoU(i,j) \leq T} H(C_{ij}, \hat{C}_{ij}) \\
& + \sum_{i=0}^{B} \sum_{c \in classes} H(p_i(c), \hat{p}_i(c)),
\end{aligned}
\tag{4.2}
$$

where B is the number of ground truth boxes, $f(w, h) = 2 - w * h$ to make the absolute displacement proportional to the box size, S is the number of grid cells along the width and the height of the feature map resulting from the considered YOLO layer, $\hat{B}$ is the number of anchor boxes, $\mathbf{1}_{ij}^{obj}$ is 1 if the anchor box j in grid cell i is responsible for predicting a bounding box

according to the ground truth and 0 otherwise, with $\mathbf{1}_{ij}^{noobj}$ being exactly the inverse, $\mathbf{1}_{ij}^{IoU(i,j)} \leq T$ is 1 if the IoU of the bounding box predicted for anchor box j in grid cell i is smaller than or equal to a predefined threshold T and 0 otherwise to ignore the prediction, and $H(y, \hat{y})$ is the binary cross-entropy function.

We implemented YOLOv3 by adapting a Keras implementation [65]. We trained the network using the official YOLOv3 weights that were pretrained on the COCO data set, in two stages: first freezing all the base network weights (Darknet53) in the first stage and then unfreezing and readjusting the weights of all layers. We used a learning rate of 0.001 in the first stage with a batch size of 32, while in the second stage we used a learning rate of 0.0001, with a decay of 0.1 in every 3 epochs without an improvement in validation loss, and a batch size of 8. As for data augmentation, we randomly employed jitter of a maximum factor of 0.1, rotations of 0, 90, 180 or 270 degrees, a hue variation between -0.05 and 0.05 with saturation and value manipulation of a maximum factor of 1.5. Finally, we set the value of the IoU ignore threshold T to 0.3. We determined all of these values empirically.

As we evaluated our detection model on the images from our test set of 221 images, our method detected 103 true positive cases of tunnels, 13 false positives and 16 false negatives, while in the remainder of the images it correctly did not detect anything. This corresponds to a recall of 86.55% and precision of 88.79% each, using an IoU threshold of only 0.3 with the ground truth, as the tunnel is difficult to localize accurately lacking clearly defined borders, and a default confidence threshold of 0.5.

We extracted the precision-recall curve from our model, which is given in Figure 4.8. In this curve, we can observe that up to a recall of 40% our model obtains a precision of 100%, while by relaxing the threshold, we can obtain up to a recall close to 100% in exchange for the precision lowering to approximately 76% at its minimum. The AP that summarizes the precision-recall curve according to the calculation method elaborated in Section 4.3.7 is 0.9571. The mean inference time per image we measured for our method was 0.0317 seconds with a standard deviation of 0.0011 seconds on a Titan V GPU after initialization.

In Figure 4.9 we show selected results from our detection model. In the upper row we can observe the true positives, while in the bottom row we show two representative cases of the false positives as well as two representative cases of the false negatives.

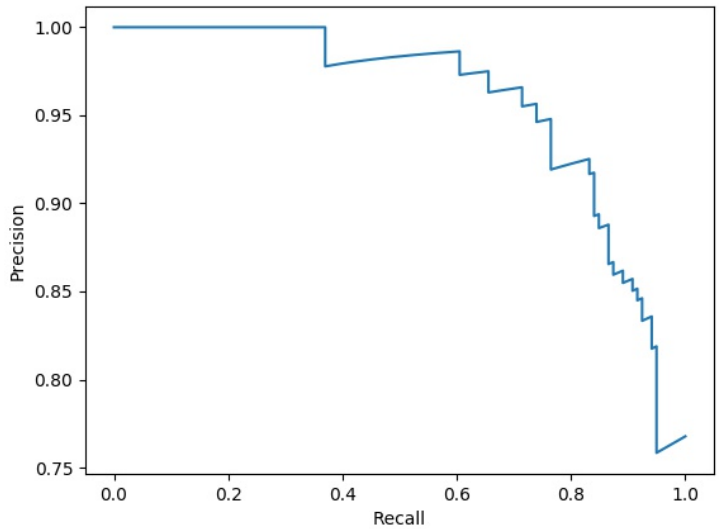


Figure 4.8: The precision-recall curve of our tunnel detection model.

4.7.3 Optical Manometry Approximation

Using our tunnel detection method from the previously described experiment, we attempted to approximate an intestinal manometry that is traditionally used to perform intestinal motility assessment using invasive procedures [83]. This experiment consists of two parts which are combined into a single visual result, namely determining the correct orientation of the capsule from the tunnel detection information and subsequently using hand-engineered feature extraction to create the manometry approximation. For this experiment, we used five CE videos.

In the first step of this method, we aimed to filter the frames where the capsule is oriented towards the anterograde pathway. Namely, only the frames where the intestinal muscles are clearly visible provide sufficient motility information, whereas the frames in which we are facing the wall would only cause noise if they were to be included our method. In order to determine how the capsule moves through the intestine and thus deduce the accurate capsule orientation at each moment, an ideal approach would be based on motion estimation techniques to deduce frame-to-frame dislocation information of the capsule. This is what we focused our first efforts on, using optical flow based on the

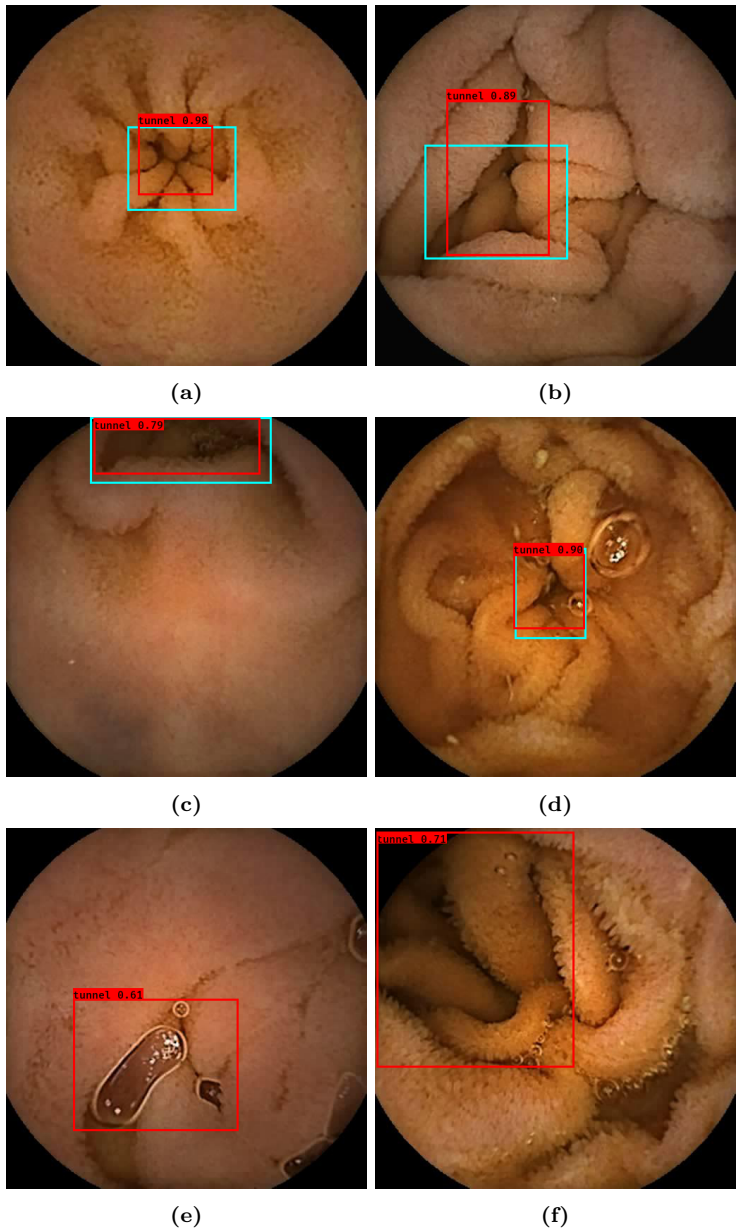
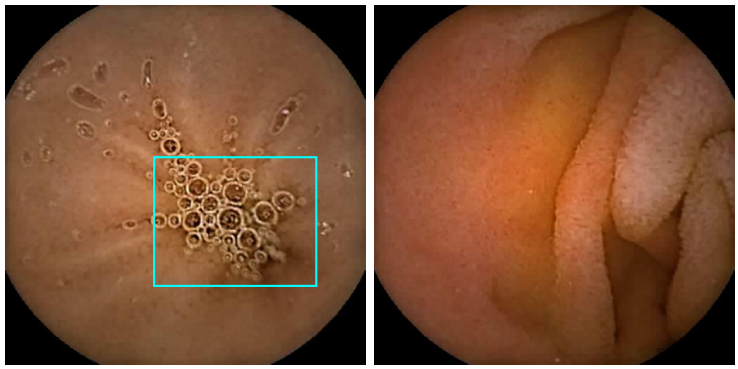


Figure 4.9: Selected examples from our method (continued on the following page). In all examples, the box as detected by our model is shown in red along with the confidence score, while the ground truth annotation is shown in blue. Figures (a), (b), (c) and (d) show true positives, while (e) and (f) show false positives.



(g) **(h)**

Figure 4.9 (cont.): Selected examples from our method (continued). Figure (g) shows a false negative, while (h) shows an example where our method correctly did not detect anything.

Lucas-Kanade method as well as AffineSIFT [100]. However, it turned out that the low framerate of 2 to 6 frames per second did not allow for such techniques to be effective, as the scenes in the frames would commonly be so far apart that they did not sufficient landmarks.

Therefore, we instead decided to filter the frames that have the approximate orientation we require for our method, i.e. facing the tunnel with the tunnel detection being located in the centre of its view, based solely on our tunnel localisation from the previous experiment. We thus discard the frames without any tunnel detections, which are considered to be oriented towards the intestinal wall. The remaining frames all contain the tunnel in its view, but the capsule may be oriented slightly sideways, due to which we first need to determine whether the tunnel detection is centred in the frame. In order to do so, we slice each frame with a tunnel detection into three equal parts both horizontally and vertically, yielding a total of nine patches of size $\frac{1}{3}w \times \frac{1}{3}h$ each as shown in Figure 4.10, where w and h refer to the width and height respectively. We then filter the frames where the centre of a bounding box from the tunnel detection was contained within the central patch and discarded those for which this criterium is not met.

From the remaining frames, we derive an approximation of a traditional manometry as we explain next. This can then be used to visually evaluate the intestinal motility instantly, while the method essentially provides a fingerprint of intestinal motility which can also be used to efficiently train machine learning algorithms to distinguish between healthy motility and motility disorders.

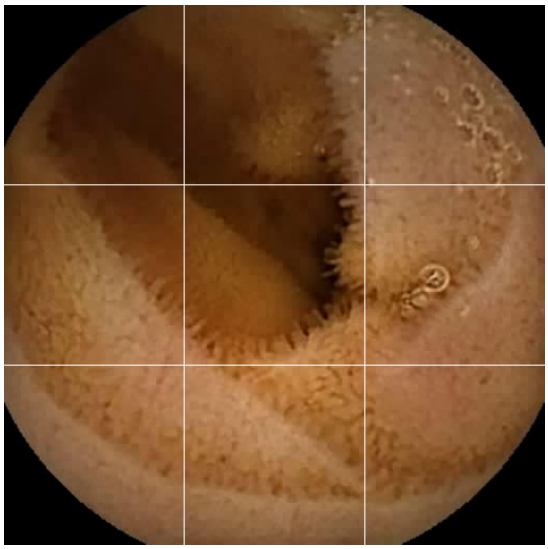


Figure 4.10: The grid from which we extracted the central patch in which we required the centre of the bounding box to be for our manometry approximation method.

To construct the approximation, for each frame we process the bounding box information. Note that under certain circumstances, e.g. false detections or tunnels that are partially closed in the middle, there may be more than one bounding box in an image. We therefore only process the bounding box that has its centre closest to the centre of the of the image.

For each frame, we then want to determine the relative size of the opening of the tunnel. As the bounding box always contains more image area than the actual tunnel, due to the nature of a bounding box, we need a more accurate method to extract the actual tunnel data from the image. One way to do this, is to obtain the segmentation directly from the CNN detector, e.g. using Mask R-CNN. However, not only is this method much more expensive in terms of resource requirements, which would make it less useful as a generic preprocessing method, but it would also require expensive mask annotations for each tunnel. Along with this, we would likely require significantly more training data to learn the correct mask segmentations, while the base model itself also requires more parameters than a light-weight method as YOLOv3.

Therefore, we instead opted to combine our tunnel detection with classical feature extraction for this purpose. An important concept in our method below is the largest inscribing circle in an area, which is used in two different

steps as we will explain in the following paragraphs. To calculate the largest inscribing circle of an area, we first need to derive a contour of the area. For this purpose, we first determine the borders of an area given by a binary image, where values of 0 correspond to the background and values of 1 correspond to the area of interest. We then construct a distance map for all pixels within the bounding box, with positive distance values for pixels within one of the detected areas and values of zero outside. From the resulting distance map, we finally derive the largest inscribing circle by finding the greatest value. The corresponding coordinates correspond to the location of the centre of the largest inscribing circle, while the value corresponds to the radius. Following, we explain our method for determining the per-frame relative tunnel size in detail. The whole procedure is also visualised in Figure 4.13.

As we already localised the tunnel within the frame, we can make assumptions about the locality of the tunnel area within the bounding box part of the image to apply segmentation techniques. We aimed to segment the darker area corresponding to the deeper part of the tunnel from the lighter area, which, in turn, would correspond to the closest contracting muscles. Therefore, we first convert the image from RGB to CIELAB in order to have the lightness of pixels separated into a single channel over which we can perform the segmentation. In order to avoid the pixels corresponding to the black frame of the image from interfering with the clustering procedure, we then extract the values corresponding to the region of interest L_r from the frame, which is the greatest informative rectangular area we can obtain from our images, as predefined by the mask image we have been using throughout our work. While clustering methods such as k-means and mean shift already proved to be useful for this purposes in our experiments, after empirical evaluation we finally opted for segmentation based on morphological active contours without edges to take advantages of the circular shape of the intestinal muscles that enclose the tunnel area. This method is a part of the family of segmentation methods of morphological snakes, which approximate contour evolution algorithms by approximating the partial differential equations used within through the successive application of a set of morphological operations [57].

The geodesic nature of this technique allows us to use the naturally present borders of the contracting muscles to segment exactly the area within the ones below a certain lightness threshold. The algorithm also guarantees that the segmented area is contained within an area of contrasting properties. This means that if a tunnel is detected and contained within our bounding box in an otherwise clean image, the tunnel can be assumed to be the darker area within the image, due to which the resulting segmented area is highly likely

to correspond to the tunnel. The particular variant we chose to use, is the approximation of active contours without edges (ACWE) [11], as in our case the contours do not always fully enclose the areas of interest and we expect different averages for the inside and outside regions of our segmentation, in which case this variant tends to work better than its alternatives.

Additionally, to steer the segmentation procedure towards the desired results, we derive an initial level set for the segmentation based on the same threshold we applied above. This part of the method only considers the part of the image within the bounding box corresponding to the detection. Namely, we create a binary image for the bounding box, in which we set all the pixels below the threshold to 1 and all the other pixels 0. To all pixels we then first apply a threshold at the value of 65, which we empirically determined to work well for this trade-off under different circumstances. For images that only contain values higher than the threshold, we instead set the threshold at $\min(L_b)+2$, where L_b corresponds to the lightness values of the bounding box, to ensure that the following steps would still derive a valid level set. From this binary image, we then derive the largest inscribing circle c_b for the greatest area. Using the properties of the derived circle, we derive a circle level set, where the relevant circle has a radius of the size of $0.5 \times \text{radius}(c_b)$, while we chose the position of the circle to be equal to the found centre, converted to its coordinates in L_r. Figure 4.11 gives a general overview of this procedure.

The segmentation itself, as well as all of the other following operations, is then performed on the lightness values of the entire area of interest, L_r, that we extracted from the frame initially. As there may be more than one area in the resulting binary segmentation image, we use the greatest one, as we assume it to be the most likely to correspond to the inner part of the tunnel under the condition of the parameters that we derived above. From the contours of this area, we then calculated the largest inscribing circle as explained previously. Some examples of original frames with the resulting largest inscribed circle of the tunnel area as an overlay, are displayed in Figure 4.12.

Finally, we convert the obtained information to a column image, where each column of pixels is constructed from a single frame in the sequence. For this, we simplified the visualisation to two colours: one colour that represents the mucosa corresponding to intestinal wall, and black, corresponding to the actual tunnel. As each column contains the same number of pixels, the absolute size of the tunnel should be converted to a value relative to the entire frame to indicate the extent of contraction. Therefore, we calculate the ratio of the tunnel size, obtained from the diameter of the largest inscribing circle, to the

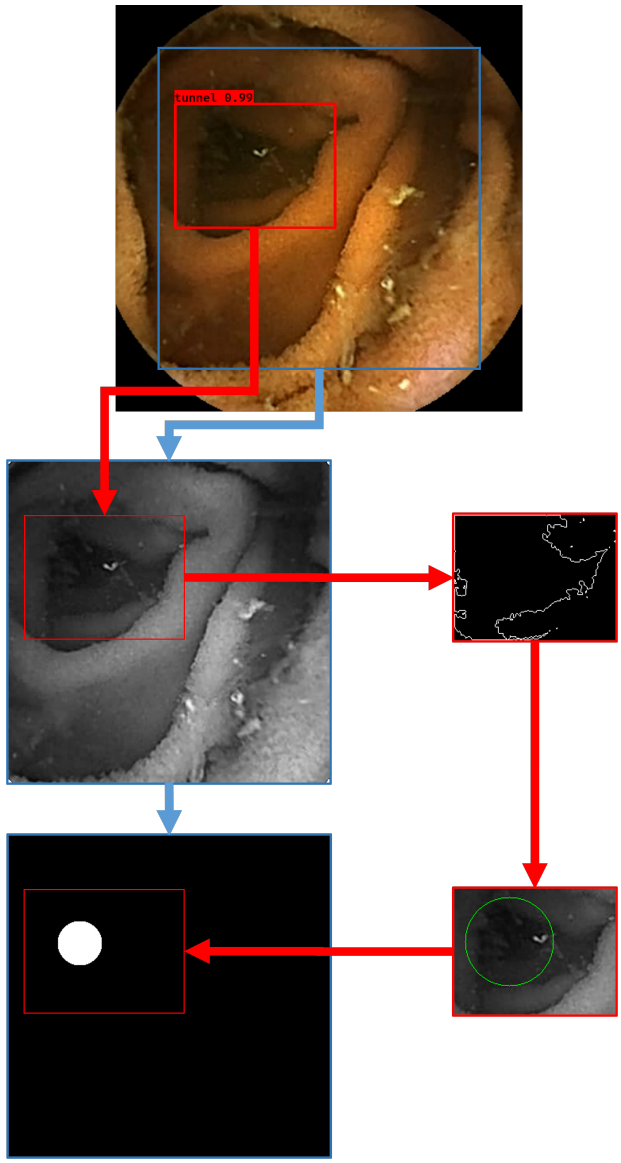


Figure 4.11: The process of deriving the initial level set for our segmentation algorithm. We first extract the lightness channel of the largest relevant area L_r as explained in the text for the main procedure. From this, we extract the values corresponding to the bounding box L_b, to which we apply a threshold, derive the contours of the area below the threshold and calculate the largest inscribing circle c_b within. Finally, we initialise the level set with the dimensions of the L_r of the image we use for the remainder of the method, the coordinates of the derived circle converted to the corresponding coordinates in L_b and half the radius of c_b.

(a) (b) (c)

Figure 4.12: Some examples of circles obtained as the largest inscribed circles by our method from the segmented areas. Figures a and b show correct results for a closed and half open tunnel respectively, while the position of the inscribed circle of Figure c shows us that the segmentation was slightly off in the presence of intestinal content.

dimensions of the image, and converted this to the number of pixels that should be coloured black in the manometry approximation.

For the five videos we used in this experiment, this led to the results provided in Figure 4.14 after processing the first 400 frames with the desired orientation. For comparison, we also attempted to visualise the original frames in a comparable column image. To do so, for each frame we divided all the pixels along the diagonal into 30 equally sized bins, after removing the pixels corresponding to the mask. We then appended a column per frame to the final image by letting each pixel correspond to the mean value of the bin at that position. Each of the images extracted from the original frames in this way is displayed above the corresponding manometry image we derived from the same frame using our method.

4.8 Discussion

In our capsule orientation estimation, we manually observed both the frames with false positives and those with false negatives in order to try to understand under what circumstances our method makes its mistakes. For both groups, we found that in the the majority of cases these situations were the same situations in which the expert could not accurately determine either the presence of the location of the tunnel and therefore hesitated in making the corresponding annotations. Examples of such frames for a false positive and a false negative are given in Figure 4.9f and Figure 4.9g respectively. However, in the case of

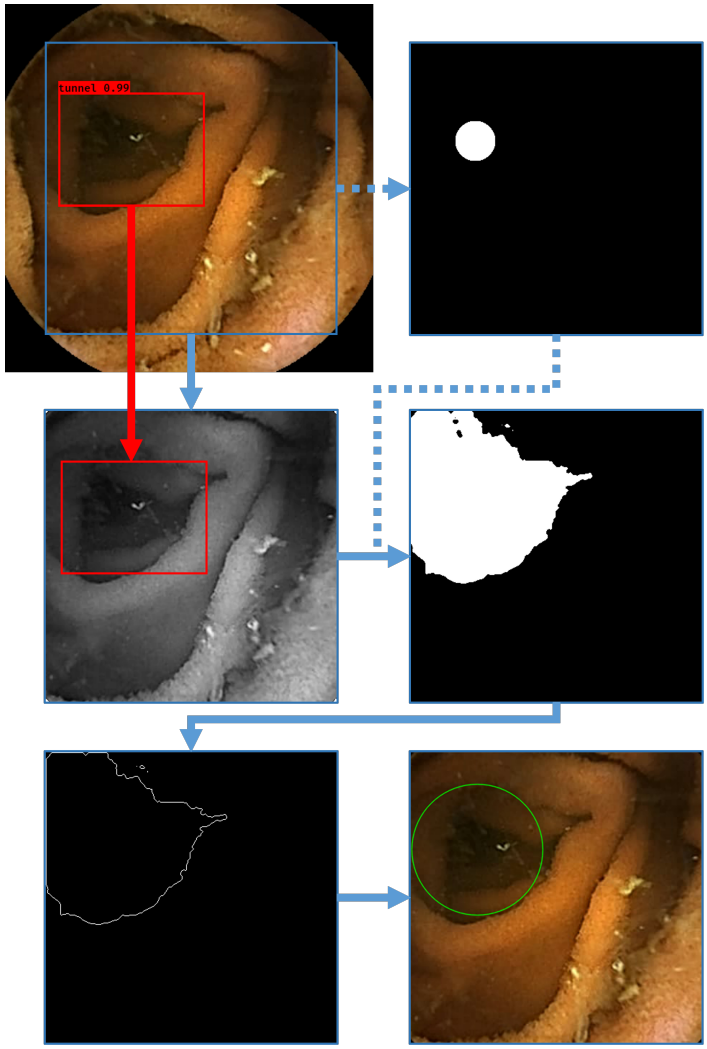


Figure 4.13: The procedure from detecting and localising the tunnel up to obtaining the inscribed circle, where the order is indicated by the red arrow. We first obtain the bounding box from our YOLOv3 detector. We then extract the greatest relevant area from the image to exclude the pixels from the black frame, convert this area from RGB to CIELAB colour space and extract the lightness channel. Using these values and the bounding box information, we then derive an appropriate initial level set for our segmenteation algorithm, a Morphological Snake approximation of ACWE, as explained in text and visualised in Figure 4.11. Finally, we perform the segmentation and calculate the largest inscribed circle from the contours of the segmented area, which gives us the values of a single column in our output manometry approximation as shown in Figure 4.14. All images are shown in original relative sizes.

.

Figure 4.14: The manometry approximations generated by our method based on visual analysis of CE procedures for the five different test videos we used (continued on following page). For comparison, for each frame we also extracted the mean colours of 30 equally sized bins along the diagonal of the original frames, excluding pixels corresponding to the black frame. Each of these is visualised above the derived manometry approximation.

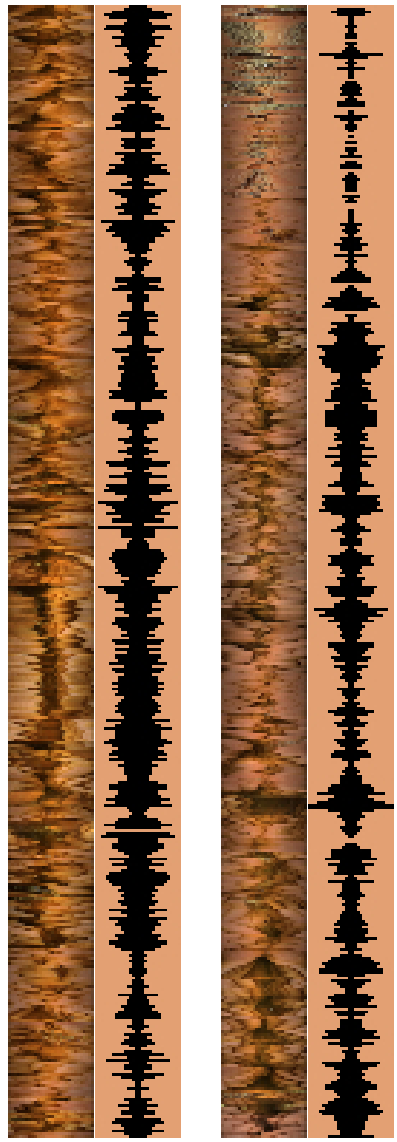


Figure 4.14 (cont.): The manometry approximations generated by our method based on visual analysis of CE procedures for the five different test videos we used (continued).

false positives, there were also cases that were more obvious to the human eye, such as Figure 4.9e.

To make a good trade-off between these values, it is important to consider the purpose. In the case of our motility visualisation, for missed tunnel frames, relating to low recall, the consequence would be that the frame gets discarded and we may miss motility information in our manometry approximation that was in fact present in the original data. This can disrupt the continuity of the result as noise is introduced. However, erroneously detected tunnels, related to low precision, would have the same consequence. Namely, this would result in the frame and the bounding box information being processed by our method, where the assumptions made for the post-processing to obtain the tunnel size, would no longer hold. Therefore, both recall and precision are important to minimise the effect of noise. For our tunnel detection model, we measured precision and recall values using the predetermined confidence threshold, which thus appears to be an acceptable trade-off.

There are two important practical aspects we have also considered in our methodology and find relevant to discuss here, even though we did not include them in our final method. The first of these was adaptation of the loss function. We initially felt this would be necessary for our method as the loss function of YOLOv3 strictly punishes any mismatch in bounding box overlap. In our case, strict bounding box overlap is not of vital importance, as it is difficult to consider where the space of the tunnel exactly starts and ends. Namely, in case of a short tunnel, e.g. before the intestine makes a curve and changes direction, there is relatively good illumination of the intestinal wall at the end of the tunnel, which makes the borders difficult to determine. We therefore considered changing the loss function both by increasing the relative weight of the confidence loss to the localization loss and by including the localization loss of only the central coordinates of the bounding box, which should correspond to the coordinates of the deepest part of the tunnel.

Another idea was to consider a different method altogether that does not perform direct bounding box regression but central coordinate regression instead from which bounding box widths and heights can subsequently be intelligently deduced [102], which at first sight appears to be more suitable to our purpose. However, through evaluation with experts we learnt that the tunnel can actually be accurately annotated through bounding box annotation. Namely, despite different grades of illumination of the tunnel depending on the visible depth, i.e. the proximity of the centre of the tunnel in non-contracted state, we will always find the relatively least illuminated part of the image, while the border around this area is practically always marked by one of the circular

muscles or folds on at least end, and the other side of the tunnel is either marked by the same end or by a intestinal wall in case of a curve. In the case of a contracted state, the centre of the contraction always has the same characteristic shape of different intestinal folds coming together. In this case, we decided the expert should annotate only the central point of this shape with a minimal amount of surrounding.

This observation of different characteristics of the tunnel in different states led us to our second idea, which was to use different detection models for tunnel detection in the contracted state and in the non-contracted state. This technique has shown promising results in related capsule endoscopy research for the colon [86]. The more different the characteristics are in a detection problem, the more complex is the training procedure and the more parameters will be necessary when attempting to include all of those characteristics in a single network. Logically, with the increase of the number of parameters and the complexity, exponentially more images will be required to train the network successfully. Also at the time of prediction, it may be more efficient in terms of computer resources to merge separate detections from two detectors than to incorporate the full detection into a single network. If a lower prediction time is required in the future, this technique may be considered. However, we did not further investigate this idea, as our method was able to sufficiently capture the complexity of the features required to recognize both distinct cases by successfully training a single, resource efficient network. We showed that our method required 0.0326 seconds to detect the tunnel with a standard variation of only 0.0019 seconds, which results in 30.67 frames per second (FPS) with small post-processing time included. Although this was on superior hardware than the available processor in endoscopic pills, it was in a non-optimised code environment, making us confident that through optimisation our method may even be used in real-time for 6 FPS, which is the maximum frame rate of current popular capsule endoscopes [59], on processors of endoscopic pills.

During our work, we became aware of a couple of limitations of our tunnel detector that we want to elaborate on here. First, even though we included frames with clear bubble formation, we did not include frames with coloured, sight-occluding intestinal content. In the presence of such intestinal content in a loose frame, it is impossible even for an expert to determine the tunnel size due to the occlusion of the relevant information. Therefore, as opposed to trying to deduce information from such frames, we suggest to discard them altogether by using our previously developed model for intestinal content detection to detect the presence of intestinal content within a tunnel bounding box.

Second, our model does not distinguish between the tunnel that the capsule has already traversed and the tunnel that eventually leads to the rectum. Namely, when the capsule rotates due to the forces acting upon it, it may be positioned backwards. In our case of motility analysis, we argue that it does not make a significant difference we expect similar movement on either side as we expect similar motion at each side of the capsule. For other purposes, however, such as capsule navigation, this may be more of an issue. In order to choose the correct tunnel in that case, we suggest the incorporation of tracking methods to attempt to track the two different tunnels throughout the procedure. Even though we argued that frame-to-frame motion estimation of the capsule is implausible considering the frame rate and the sudden movements of the capsule, we hypothesise that it is plausible to track the different features observed when the capsule faces either way. This hypothesis would however need to be investigated first, although future development of active capsules may make it possible for capsules to be controlled and steered at a constant rate. In the latter case, additionally to the possibility of incorporating accurate tracking mechanisms through this method to track the right tunnel, we could also improve detection rates even further, as we would be able to relate the tunnel in the contracted state to the tunnel in the non-contracted state in an earlier or later frame.

Finally, we consider that an important limitation of our method is the case of detecting two tunnels in a single frame. Often, one of the tunnels is then likely to be a false detection. For our purpose of intestinal motility analysis, we have already discussed this issue and our considerations, but if our tunnel detection method is used for other purposes such as navigation purposes, it is important to consider that sometimes there can in fact be two tunnels in a frame. Namely, this can be due to a rare condition that occurs in patients who have had previous bowel surgery and in this case, normally either of the tunnels will lead to the rectum. However, in certain cases one of the tunnels may in fact be a dead end, as is the case with the appendix. In such a case, making the wrong choice could lead to health consequences and high discomfort for the patient, as it could theoretically induce an appendix infection. Therefore, for such purposes, it should be possible to extend this method with a mechanism to make a more informed choice for right tunnel, as experts can also distinguish the appendix from the usual tunnel in endoscopic images.

Concerning our manometry approximation, we believe that it still has certain limitations relating to the relative tunnel size. One of these limitations is in case we are close to an intestinal curve and thus deal with shallow tunnels, which was already discussed as an issue in determining the correct tunnel

size both for our tunnel detection model and for human evaluation in the annotation procedure. Namely, our approach may not be able to determine the correct tunnel size close to the a curve in the intestine, where the tunnel is likely to be shallow, which could result in a small tunnel size in our approach in spite of relaxed muscles. To improve this, we believe that it would be useful to train a model with sufficient data to distinguish not only between tunnel and wall, but also between tunnel in state of contraction and in state of relaxation. One could then base the value of the threshold in determining the initial level set for the algorithm on the type of tunnel, i.e. make a different choice for a contracted tunnel than for an open tunnel.

Additionally, upon inspection of the largest inscribed circle results from which we derived the relevant information for the manometry approximation, we found that there was only one obvious cause for inscribed circles being significantly off either in position or size, which was the presence of intestinal content. Independently from the tunnel detection, our method does not perform well in the presence of intestinal content, which is often in front of the tunnel and closer to the camera, resulting in higher lightness values. This is especially an issue if the intestinal content is dense, or when light reflections from bubbles cause our method to segment darker tunnel areas areas contained within small bubbles separately. While two of the images in Figure 4.12 show results exactly as desired, both the position and the size of the circle in Figure 4.12c show how the segmentation for that image did not include the whole tunnel within the largest inscribing circle due to the segmentation being influenced by the intestinal content. In future lines of work, a possible solution here would be to use an intestinal content detection model such as the one introduced in Chapter 3 in order to filter out those frames. This would, however, disturb the continuity of the manometry approximation.

Finally, the latter leads us to the last important limitation we found for our method, which is in the outliers in the manometry approximation. Apart from the issues discussed above that could lead to of incorrectly determined tunnel sizes or to false detections from our tunnel detection approach, these outliers are due to frames that were filtered out for not having the tunnel in the centre of the image as a consequence of rotation of the capsule. We believe that this issue can be diminished in future work by discarding more frames in between to align the tunnel frames more carefully. An different or additional approach could be to investigate further post-processing to the resulting approximation, e.g. through fitting a function to the individual or combined contractions and openings, which could also help to fill in missing data from discarded frames for relatively small gaps.

The effect of these limitations can be observed in Figure 4.14. Namely, we can clearly observe the effect of discontinuities in the manometry approximation due to frames discarded while facing the intestinal wall, which commonly align with discontinuities in the original frames, thus showing that these are due to discarded frames rather than incorrectly derived tunnel sizes. Additionally, we can observe incorrectly derived tunnel sizes, which show up as outliers in the first, second and last video especially. In the majority of these cases, we found they were due to the presence of intestinal content upon closer inspection. We can also clearly observe the mentioned effect of shallow tunnels in our results through the longer segments with shallow tunnels in the second, third and fifth video. This pattern is can clearly be seen from the colours in the original frames as well. Regardless, within most of these segments, tunnel sizes within a segment are commonly within a similar range and the pattern of intestinal motility, although reduced, still appears to be there, and can be accounted for by observers or processing algorithms. In most cases, we thus observe that the information given by the derived manometry approximations align correctly with the information we can deduce from the original, while it filters and displays the motility information in such a way that it is significantly better observed both by human observers and processing algorithms than the column image of the original frames. Therefore, despite the limitations, we believe that our method can be valuable for its intended purpose of computer-aided assistance in motility analysis.

We are aware that a manometric recording for intestinal motility is commonly performed by introducing different substances and carefully measuring the intestinal motility response to each of these. In the case of capsule endoscopy, we do not have this level of control, while at the same time the visibility would be compromised in case of digestion. As we have both seen and discussed in Chapter 3, it is vital to the degree of visibility of the visual recording system of capsule endoscopy that the procedure is used in fasting state only. For patients who need a more thorough motility analysis, the method presented in this paper may thus be insufficient. Nonetheless, while our method does not explicitly detect intestinal contractions, in contrast to an earlier study [90], we believe the method presented here extracts both more detailed and more relevant information for computer-aided analysis of intestinal motility, and provides an excellent base for future methods that do want to detect specific events related to intestinal motility.

Chapter 5
Conclusion

This thesis project took placing during a phase in which machine learning research was transitioning from hand-crafted feature extraction to deep learning techniques using CNNs. While CNN theory is in fact aimed at generic parameter optimisation and often requires little domain knowledge from the researcher to obtain decent results, the features that are extracted within are often poorly understood, even though visualisation techniques exist and are providing us with hopeful information to what would otherwise be considered a *black box*. Today, we can say that CNNs have surpassed traditional machine learning techniques in a wide range of fields, despite the genericness of the method and extensive search space. Nonetheless, we think it is interesting to keep traditional machine learning techniques in mind, as features extracted through such methods can be combined with CNNs, while knowledge of how they work can also help in reasoning about CNN architectures and constructing better performing CNN methods.

In this work, we therefore investigated and implemented both traditional machine learning techniques and deep learning techniques. Using these either individually or combined, we aimed to contribute to the field of capsule endoscopy, which allows for the visualisation of the entire small intestine for disease diagnosis in a minimally invasive manner. In close cooperation with researchers and gastroenterologists from hospital La Fe in Valencia, we identified important issues in CE that are underrepresented in contemporary research where there is still significant advancement to be made, other than in automatic detection of pathologies. We identified two objectives that we aimed to achieve in this work, which are both related to optimising the information extraction from capsule endoscopy through often overlooked manners. Namely, our main objective was to develop a tool for automatic, objective assessment of the degree of visibility, or cleanliness, of the intestinal mucosa, as there is still no consensus on the optimal patient preparation in the medical field due to which CE procedures often still need to be repeated as a consequence of a lack of visibility. Secondary to the main objective, we defined an additional objective to automatically determine the capsule orientation throughout a CE procedure for motility analysis.

Through our main objective, we aimed to provide medical researchers with a means for accurate and objective evaluation of the effect of different preparation methods for CE patients. This would allow them to reach consensus on the optimal method, which has not been possible so far due to conflicting results in studies in the absence of such a means. For this purpose, we developed an intestinal content detection algorithm to form the core of our method. As the intestinal content detection is the most significant part of the evaluation, this is what we focused most of our research work on. We started off with carefully designed traditional machine learning methods, compared their performance to fine-tuned CNN architectures and eventually decided to find the solution for this objective completely within the field of CNN techniques.

In order to further improve performance compared to fine-tuned architectures, we eventually designed a new CNN architecture tailored to our problem domain, with which we reached both improved efficiency and higher accuracy compared to fine-tuned architectures. The core of our proposed method is a model based on a CNN architecture we designed, capable of classifying image patches into intestinal content or clean mucosa, which was inspired by the architecture presented in [37]. For this model we obtained a high accuracy of 95.23%, surpassing the accuracy of the model we trained on the popular VGG-16 architecture using exactly the same data, while we achieved a significantly lower prediction time and decreased number of parameters. We also obtained significantly higher accuracy for this model than for the one based on the original architecture.

Using this model for patch-based classification at the core, we finally implemented the entire method for complete cleanliness evaluation of CE videos, by estimating pixel probabilities from the patch probabilities and visualising these intuitively, while also determining the average probability of a pixel of corresponding to intestinal content and finally converting this to an evaluation score on a discrete scale with four categories. In this way, the proposed method is capable of automatically evaluating the frames of CE videos on a scale that is meant to be equivalent to the cleanliness evaluation score proposed in [5]. We then validated our method in a clinical setting, evaluating the scores thus assigned to 854 frames extracted from 30 different CE videos using a 5-fold cross-validation procedure for threshold adjustment, we found acceptable agreement with each of the human specialists, despite strong disagreement between our method and both specialists on frames that showed bleeding in one fold in particular, where our method correctly did not detect the blood as intestinal content. In a three-way comparison, using the two-way mixed, absolute agreement-based intraclass correlation coefficient for

single rater reliability, we also showed that the value found when including our method did not move outside the 95% confidence interval found for both specialists alone.

Despite promising results in the validation of our method, we have observed and discussed some current limitations of our method, most of which we believe we can overcome by adapting our method in future lines of study. Namely, it may be interesting to investigate the influence of the relative location of detected intestinal content in the cleanliness as perceived by humans. Additionally, we aim to further optimise our thresholds on new, more extensive data sets using video material being currently collected from different medical centres world-wide, to assure that the thresholds are averaged out over a wide cohort of medical specialists to obtain a satisfactory agreement with each one of them individually, and to simultaneously further extend the validation of our method in a clinical setting involving a greater number of specialists from different institutions.

To accomplish our secondary objective, we first developed a method to determine the capsule orientation. For this purpose, we implemented an underlying method to automatically detect the *tunnel*, which is the term we use to refer to the anterogade patheway, both in the state of contracted muscles and in the state of relaxed muscles. Using the YOLOv3 R-CNN-based object detector, pretrained on the COCO data set [52], in combination with an annotated private dataset that included the tunnel in all states of contraction, we managed to achieve a high recall and precision of 86.55% and 88.79% each at the default threshold of 0.3 IoU. The average precision (AP), which relates to the area under the precision-recall curve, was as high as 0.9571, indicating a robust classifier as it returns true tunnel detections with high confidence before starting to return more false detections at lower confidence thresholds.

While the tunnel detection and capsule orientation determination is useful for multiple purposes as discussed in the previous chapter, we used the results to construct an approximation of an intestinal manometry. Intestinal manometry is an invasive technique to record the activity of the intestinal muscles, which is essentially displayed as a graph of muscle activity over time. This in turn gives physicians information about the intestinal motility of a patient and possible disorders thereof. We instead proposed a method to perform motility analysis using only a capsule endoscopy procedure. Our method is built on the tunnel detection method to align the frames where the capsule is oriented towards the tunnel and extracted relevant features from the detection bounding boxes to estimate the relative tunnel size. Subsequently, we used this information

to construct a visual equivalent of an intestinal manometry as visualised in Figure 4.14.

For future lines of study, we consider our tunnel detection method may be improved by training a deeper R-CNN-based architecture, which should at the same time be combined with more training data, while more training data could also allow the model to be trained to distinguish between contracted tunnels and open tunnels. Additionally, in the case of future capsule endoscopes allowing for a higher recording frame rate, we think it could be interesting to incorporate long short-term memory (LSTM) or the more recent Transformer architectures for frame-to-frame tracking. Unfortunately, in our experiments we have observed that frame-to-frame tracking in videos of the currently available capsule endoscopes appears to be more complicated than in other settings due to the low frame rates in combination with sudden movements.

Concerning our manometry approximation method that we built upon the tunnel detection, we consider it should be validated on patients with and without diagnosed motility disorders to determine whether physicians can properly distinguish between both cases using solely our method. Additionally, despite the limitation of our method of not being able to feed the patient different substances during the recording, our method should be compared against actual manometry procedures recorded for the same patient in future work. Finally, we consider that a more sohpisticated approach towards false detections as well as a method of reducing noise originating from the respective exclusion or inclusion of such frames, could improve the results of the manometry approximation.

While medical research and technical research often go separate directions, we were fortunate to be able to collaborate closely with medical doctors from hospital La Fe in the context of the WiBEC project. Through this cooperation, we have been able to conduct research on problems surrounding the still novel technique of capsule endoscopy that are under-represented in contemporary research. As a result, we have been able to produce tangible results with our intestinal cleanliness evaluation tool, which is currently being used in a large-scale medical study that aims to achieve consensus on the issue of adequate patient preparation for CE procedures, in which we actively participate to acquire the data, process the data through our tool and organise the results in a completely automatic and anonymous manner with encrypted data transfers.

Publications

Journal Papers

Noorda, R.; Nevárez, A.; Colomer, A.; Pons Beltrán, V.; Naranjo Ornedo, V. Automatic evaluation of degree of cleanliness in capsule endoscopy based on a novel CNN architecture. *Scientific Reports* **10**, 17706 (2020).

Noorda, R.; Naranjo Ornedo, V.; Pons, V. (2017). Performance of Common Clustering Methods in Segmenting Vascular Pathologies in Capsule Endoscopy Images. *International Journal of Computer Assisted Radiology and Surgery.* **12**(1):S22-S23.

International Conferences

Noorda, R.; Nevárez, A.; Colomer, A.; Naranjo, V.; Pons Beltrán, V. (2020). Automatic Detection of Intestinal Content to Evaluate Visibility in Capsule Endoscopy. In *IEEE 13th International Symposium on Medical Information Communication Technology.*

Pons Suñer, P.; **Noorda, R.**; Nevárez, A.; Colomer, A.; Pons Beltrán, V.; Naranjo, V. (2019). Design and Development of an Automatic Blood Detection System for Capsule Endoscopy Images. In *Intelligent Data Engineering and Automated Learning – IDEAL 2019.* Lecture Notes in Computer Science, vol 11872.

National Conferences

Pons Suñer, P.; **Noorda, R.**; Naranjo, V.; Nevárez Heredia, A.; Pons Beltrán, V. (2018). Diseño y desarrollo de un sistema para la detección automática de sangre en imágenes de cápsula endoscópica. VISILAB. 257-260.

Nevárez Heredia, Andrea; **Noorda, R.**; Naranjo, V.; Pons Beltrán, Vicente. (2018). Validación de una herramienta automática para la evaluación

del grado de limpieza en estudios de cápsula endoscópica. In *41 Congreso de la Sociedad Española de la Endoscopía Digestiva.* Revista Española de Enfermedades Digestivas.

Nevárez Heredia, Andrea; **Noorda, R.**; Naranjo, V.; lonso-Lázaro, N.; Campos, M.; Pons Beltrán, Vicente. Endoclaen: Analizador Automático de Imagen para la Evaluación del Grado de Limpieza de los Procedimientos de Cápsula Endoscópica. In *40 Congreso de la Sociedad Española de la Endoscopía Digestiva.* Revista Española de Enfermedades Digestivas.

Bibliography

[1] Vincent Andrearczyk and Paul F. Whelan. Using filter banks in convolutional neural networks for texture classification. *Pattern Recognition Letters*, 84:63–69, Dec 2016.

[2] Vincent Andrearczyk and Paul F. Whelan. Deep learning in texture analysis and its application to tissue image classification. In *Biomedical Texture Analysis*, pages 95–129. Elsevier, 2017.

[3] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. In *International conference on machine learning*, pages 584–592. PMLR, 2014.

[4] Khayrul Bashar, Kensaku Mori, Yasuhito Suenaga, Takayuki Kitasaka, and Yoshito Mekada. Detecting informative frames from wireless capsule endoscopic video using color and texture features. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI 2008)*, pages 603–610, Berlin, Heidelberg, 2008. Springer.

[5] Vicente Pons Beltrán, Cristina Carretero, Begoña Gonzalez-Suárez, Iñaqui Fernández-Urien, and Miguel Muñoz-Navas. Intestinal preparation prior to capsule endoscopy administration. *World Journal of Gastroenterology: WJG*, 14(37):5773, 2008.

[6] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*. ACM Press, 1992.

[7] Byungkyu Kim, Sukho Park, Chang Yeol Jee, and Seok-Jin Yoon. An earthworm-like locomotive mechanism for capsule endoscopes. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2997–3002, 2005.

[8] Giuseppe Cardillo. Cohen's kappa. `https://www.github.com/dnafinder/Cohen`, 2020.

[9] F. Carpi, N. Kastelein, M. Talcott, and C. Pappone. Magnetically controllable gastrointestinal steering of video capsules. *IEEE Transactions on Biomedical Engineering*, 58(2):231–234, 2011.

[10] Gavin C. Cawley and Nicola L. C. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11(70):2079–2107, 2010.

[11] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.

[12] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gerard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 192–204, San Diego, California, USA, 09–12 May 2015. PMLR.

[13] G. Ciuti, R. Donlin, P. Valdastri, A. Arezzo, A. Menciassi, M. Morino, and P. Dario. Robotic versus manual control in magnetic steering of an endoscopic capsule. *Endoscopy*, 42(2):148–152, Dec 2009.

[14] Mayo Clinic. Capsule endoscopy. `https://www.mayoclinic.org/tests-procedures/capsule-endoscopy/about/pac-20393366`, Jul 2019.

[15] Jacob Cohen. Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213, 1968.

[16] Wikipedia contributers. Given Imaging. `https://en.wikipedia.org/wiki/Given_Imaging`, Apr 2021.

[17] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[18] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, Dec 1989.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR09*, 2009.

[20] Timothy Dozat. Incorporating Neseterov momentum into Adam. `https://cs229.stanford.edu/proj2015/054_report.pdf`, 2016.

[21] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Jun 2014.

[22] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, Sep 2009.

[23] Adam D Farmer, S Mark Scott, and Anthony R Hobson. Gastrointestinal motility revisited: The wireless motility capsule. *United European Gastroenterology Journal*, 1(6):413–421, Dec 2013.

[24] Giovanni Gallo and Alessandro Torrisi. Lumen detection in endoscopic images: a boosting classification approach. *International Journal On Advances in Intelligent Systems*, 5(1), 2012.

[25] Sharon Gillson. Gastrointestinal motility disorders: Digestive and non-digestive causes. `https://www.verywellhealth.com/gastrointestinal-motility-disorders-1741817`.

[26] Ross Girshick. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Dec 2015.

[27] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Jun 2014.

[28] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[29] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Maximum Likelihood Estimation*, chapter 5, page 129. MIT Press, 2016. `http://www.deeplearningbook.org`.

[30] Rachel M. Gwynne, E. A. Thomas, S. M. Goh, H. Sjövall, and J. C. Bornstein. Segmentation induced by intraluminal fatty acid in isolated guinea-pig duodenum and jejunum. *The Journal of Physiology*, 556(2):557–569, Apr 2004.

[31] Mark Berner Hansen. Small intestinal manometry. *Physiological research*, 51 6:541–56, 2002.

[32] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Oct 2017.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[34] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[35] D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, 195(1):215–243, Mar 1968.

[36] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[37] Xiao Jia and Max 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)*, pages 639–642. IEEE, 2016.

[38] H. Keller, A. Juloski, H. Kawano, M. Bechtold, A. Kimura, H. Takizawa, and R. Kuth. Method for navigation and control of a magnetically guided capsule endoscope in the human stomach. In *2012 4th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 859–865, 2012.

[39] Jutta Keller, Christiane Fibbe, Ulrich Rosien, and Peter Layer. Recent advances in capsule endoscopy: development of maneuverable capsules. *Expert Review of Gastroenterology & Hepatology*, 6(5):561–566, 2012.

[40] P. C. Khun, Zhang Zhuo, Liang Zi Yang, Li Liyuan, and Liu Jiang. Feature selection and classification for wireless capsule endoscopic frames. In *2009 International Conference on Biomedical and Pharmaceutical Engineering*, pages 1–6, Dec 2009.

[41] Amir Klein, Moshe Gizbar, Michael J Bourke, and Golo Ahlenstiel. Validated computed cleansing score for video capsule endoscopy. *Digestive Endoscopy*, 28(5):564–569, 2016. JGES-DEN-2015-5443.R2.

[42] Terry K. Koo and Mae Y. Li. A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *Journal of Chiropractic Medicine*, 15(2):155–163, Jun 2016.

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[44] Edwin J. Lai, Audrey H. Calderwood, Gheorghe Doros, Oren K. Fix, and Brian C. Jacobson. The Boston bowel preparation scale: a valid and reliable instrument for colonoscopy-oriented research. *Gastrointestinal Endoscopy*, 69(3):620–625, Mar 2009.

[45] Hei Law and Jia Deng. CornerNet: Detecting objects as paired keypoints. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[46] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan-Kaufmann, 1990.

[47] Seung-Hwa Lee, Young-Kyu Park, Sung-Min Cho, Joon-Koo Kang, and Duck-Joo Lee. Technical skills and training of upper gastrointestinal endoscopy for new beginners. *World Journal of Gastroenterology: WJG*, 21(3):759, 2015.

[48] Fei-Fei Li, Andrej Karpathy, and Justin Johnson. CS231n convolutional neural networks for visual recognition. `https://cs231n.github.io/convolutional-networks/`, 2020.

[49] Panpeng Li, Ziyun Li, Fei Gao, Li Wan, and Jun Yu. Convolutional neural networks for intestinal hemorrhage detection in wireless capsule endoscopy images. In *Multimedia and Expo (ICME), 2017 IEEE International Conference on*, pages 1518–1523. IEEE, 2017.

[50] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Light-head R-CNN: In defense of two-stage object detector. *arXiv preprint arXiv:1711.07264*, 2017.

[51] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, Feb 2020.

[52] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[53] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[54] Michael London and Michael Häusser. Dendritic computation. *Annual Review of Neuroscience*, 28(1):503–532, Jul 2005.

[55] O Haji Maghsoudi, Alireza Talebpour, Hamid Soltanian-Zadeh, Mahdi Alizadeh, and H Asl Soleimani. Informative and uninformative regions detection in WCE frames. *Journal of Advanced Computing*, 3(1):12–34, 2014.

[56] Mia L. Manabat. Intestinal motility disorders. `https://emedicine.medscape.com/article/179937-overview`, Jan 2022.

[57] Pablo Marquez-Neila, Luis Baumela, and Luis Alvarez. A morphological approach to curvature-based evolution of curves and surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(1):2–17, Jan 2014.

[58] K McLaren. XIII—The development of the CIE 1976 (L* a* b*) uniform colour space and colour-difference formula. *Journal of the Society of Dyers and Colourists*, 92(9):338–341, 1976.

[59] Medtronic. PillCam™ SB 3 system. `https://www.medtronic.com/covidien/en-us/products/capsule-endoscopy/pillcam-sb-3-system.html`.

[60] Philip W. Mewes, Dominik Neumann, Oleg Licegevic, Johannes Simon, Aleksandar Lj. Juloski, and Elli Angelopoulou. Automatic region-of-interest segmentation and pathology detection in magnetically guided capsule endoscopy. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 141–148. Springer, 2011.

[61] C. Alexander Mosse, Timothy N. Mills, Mark N. Appleyard, Srinathan S. Kadirkamanathan, and C.Paul Swain. Electrical stimulation for propelling endoscopes. *Gastrointestinal Endoscopy*, 54(1):79–83, Jul 2001.

[62] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, Jul 2002.

[63] Timo Ojala, Matti Pietikainen, and David Harwood. Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. In *Proceedings of 12th international conference on pattern recognition*, volume 1, pages 582–585. IEEE, 1994.

[64] Vicente Pons Beltrán, Begoña González Suárez, Cecilia González Asanza, Enrique Pérez-Cuadrado, Servando Fernández Diez, Iñaqui Fernández-Urién, et al. Evaluation of different bowel preparations for small bowel capsule endoscopy: A prospective, randomized, controlled study. *Digestive Diseases and Sciences*, 56(10):2900–2905, Oct 2011.

[65] qqwweee. qqwweee/keras-yolo3. `https://github.com/qqwweee/keras-yolo3/`, Jul 2018.

[66] Marco Quirini, Arianna Menciassi, Sergio Scapellato, Paolo Dario, Fabian Rieber, Chi-Nghia Ho, Sebastian Schostek, and Marc Oliver Schurr. Feasibility proof of a legged locomotion capsule for the GI tract. *Gastrointestinal Endoscopy*, 67(7):1153–1158, Jun 2008.

[67] Joseph Redmon. Darknet. `https://github.com/pjreddie/darknet/`, 2020.

[68] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[69] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jul 2017.

[70] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2019.

[71] Russell D. Reed and Robert J. Marks. *MLP Representational Capabilities*, chapter 4, page 31. The MIT Press, 1999.

[72] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, Jun 2017.

[73] Sebastian Ruder. An overview of gradient descent optimization algorithms. `https://ruder.io/optimizing-gradient-descent/index.html`, Mar 2020.

[74] 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. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[75] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009.

[76] Sebastian Schostek, Melanie Zimmermann, Jan Keller, Mario Fode, Michael Melbert, Ruediger L. Prosst, Thomas Gottwald, and Marc O. Schurr. Pre-clinical study on a telemetric gastric sensor for recognition of acute upper gastrointestinal bleeding: the "HemoPill monitor". *Surgical Endoscopy*, 34(2):888–898, May 2019.

[77] Santi Segui, Michal Drozdzal, Fernando Vilariño, Carolina Malagelada, Fernando Azpiroz, Petia Radeva, and Jordi Vitria. Categorization and segmentation of intestinal content frames for wireless capsule endoscopy. *IEEE Transactions on Information Technology in Biomedicine*, 16(6):1341–1352, 2012.

[78] Jonathan D. Siegel and Jack A. Di Palma. Medical treatment of constipation. *Clinics in Colon and Rectal Surgery*, 18(02):76–80, May 2005.

[79] Julius Sim and Chris C. Wright. The kappa statistic in reliability studies: Use, interpretation, and sample size requirements. *Physical Therapy*, 85(3):257–268, Mar 2005.

[80] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[81] Mark Skilton and Felix Hovsepian. *Advanced Neural Networks*, pages 159–187. Springer International Publishing, Cham, 2018.

[82] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[83] V. Stanghellini, R. Cogliandro, L. Cogliandro, R. De Giorgio, G. Barbara, B. Salvioli, and R. Corinaldesi. Clinical use of manometry for the diagnosis of intestinal motor abnormalities. *Digestive and Liver Disease*, 32(6):532–541, Aug 2000.

[84] Z. Sun, B. Li, R. Zhou, H. Zheng, and M. Q. H. Meng. Removal of non-informative frames for wireless capsule endoscopy video segmentation. In *2012 IEEE International Conference on Automation and Logistics*, pages 294–299, Aug 2012.

[85] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[86] Nima Tajbakhsh, Suryakanth R. Gurudu, and Jianming Liang. Automated polyp detection in colonoscopy videos using shape and context information. *IEEE Transactions on Medical Imaging*, 35(2):630–644, Feb 2016.

[87] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[88] Jasper R. R. Uijlings, Koen E. A. van de Sande, Theo Gevers, and Arnold W. M. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[89] Math Vault. Chain rule for derivative - the theory. `https://mathvault.ca/chain-rule-derivative/`, Sep 2020.

[90] F. Vilariño, P. Spyridonos, F. DeIorio, J. Vitria, F. Azpiroz, and P. Radeva. Intestinal motility assessment with video capsule endoscopy: Automatic annotation of phasic intestinal contractions. *IEEE Transactions on Medical Imaging*, 29(2):246–259, Feb 2010.

[91] F. Vilariño, P. Spyridonos, O. Pujol, J. Vitria, and P. Radeva. Automatic detection of intestinal juices in wireless capsule video endoscopy. In *18th*

International Conference on Pattern Recognition (ICPR'06), volume 4, pages 719–722, 2006.

[92] Qian Wang, Ning Pan, Wei Xiong, Heng Lu, Nan Li, and Xijing Zou. Reduction of bubble-like frames using a RSS filter in wireless capsule endoscopy video. *Optics & Laser Technology*, 110:152–157, 2019.

[93] Matthijs J. Warrens. Conditional inequalities between Cohen's kappa and weighted kappas. *Statistical Methodology*, 10(1):14–22, 2013.

[94] Ian L. Weatherall and Bernard D. Coombs. Skin color measurements in terms of CIELAB color space values. *Journal of Investigative Dermatology*, 99(4):468–473, Oct 1992.

[95] Paul J. Werbos et al. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[96] Wikipedia contributors. Backpropagation — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Backpropagation&oldid=897855738`, 2020. [Online; accessed 04-Sep-2020].

[97] Shuang Wu, Guanrui Wang, Pei Tang, Feng Chen, and Luping Shi. Convolution with even-sized kernels and symmetric padding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 1194–1205. Curran Associates, Inc., 2019.

[98] L. Xu. Loss function in ML. https://lucasxlu.github.io/blog/2018/07/24/ml-loss/, Jul 2020.

[99] Hironori Yamamoto and Kentaro Sugano. A new method of enteroscopy–the double-balloon method. *Canadian Journal of Gastroenterology*, 17(4):273–274, 2003.

[100] Guoshen Yu and Jean-Michel Morel. ASIFT: An algorithm for fully affine invariant comparison. *Image Processing On Line*, 1:11–38, 2011.

[101] X. Zabulis, A. A. Argyros, and D. P. Tsakiris. Lumen detection for capsule endoscopy. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3921–3926, 2008.

[102] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.

Appendix A

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a branch of artificial neural networks (ANNs), the design of which was largely inspired by work done on the functioning of the mammal brain by Hubel and Wiesel as early as 1968 [35]. Due to this, many of the terms have brain analogies. Although we will briefly introduce them, we will not expand on them in detail as we consider it to be outside the scope of this thesis. Namely, comprehension of the biological analogies is not essential to understand the theory and the mathematics behind neural networks that we will focus on. Additionally, ANN and CNN models are highly simplified compared to actual current neuroscience, which has advanced significantly since the aforementioned work, finding among other things that there are many different types of neurons and that synapses do not just have a single weight as we will detail out below for CNNs, but are on themselves complex non-linear systems [54]. Nonetheless, CNN models are powerful tools in the performance complex machine learning tasks. Although CNNs obviously have a lot of common ground with general ANN theory, we will not explain general neural network theory more in detail than required to properly understand the concept of CNNs in the context of image processing, or more specifically, our CE image patches. We will, however, sometimes resort to basic ANN theory to explain certain CNN concepts, as this allows us to simplify those concepts on lower-dimensional data. Another term for an ANN is Multi-Layer Perceptron (MLP).

Apart from their similarities with other forms of ANNs, CNNs distinguish themselves with the introduction of important new concepts designed specifically for the processing of images, pioneered by [46]. The most characteristic of these concepts is the one it derives its popular name from, namely the convolution operation (although this is in fact a correlation, as we will discuss in Section A.3). Processing image data through basic ANNs would be unnecessarily expensive in terms of operations and computer resources considering the number of pixels a single image usually contains already with lower dimensions, but even more so with the image dimensions that are common nowadays. The assumption of the input data being image data, however, allows for the implementation of concepts that make sense for image data, while they may not for generic data, such as the sharing of weights between input pixels at

different locations of the input image as introduced in the work by LeCun et al. [46]. This constraint on the weights is, in fact, the way the convolution operation is implemented in the network. Thanks to these concepts, not only can the network size and computation times be significantly reduced in comparison with processing image data through basic ANNs, but at the same the performance is often improved.

As CNNs, just like ANNs, can be considered as universal function approximaters, there is essentially no limit to the number and ranges of output values they can be predict. This makes them applicable both for regression, where the output value(s) are continuous, and classification tasks, where the output value(s) are discrete. In this appendix, however, we only consider CNNs for classification, unless specified otherwise.

A.1 Learning Procedure

In general, neural networks can be considered function approximators. In fact, it has been proven that a neural network can, when certain conditions are met, uniformly approximate any continuous function [18]. Whether in the end we manage to make them in fact approximate a desired function, depends largely on the configuration of the network and on its parameters or *weights*. These may be learnt through the training procedure such that the neural network can adapt those to the problem at hand as much as possible. The configuration of a neural network mainly depends on the architecture and on its hyperparameters that are used during training, and therefore we will thus first discuss the architecture of CNNs in Section A.2. The architecture is made up out of layers of different types. In ANNs these types are mainly distinguished by their connectivity with other layers or by their operations, and in CNNs this is essentially no different. However, the layer operations are focused on 2-dimensional input data, such as images, which may make the operations more intuitive and allow for a better visual imagination, while they abstract the fact that such operations are essentially implemented as a combination of specific connectivity and restrictions on updating the weights and using functions that are specifically useful for 2-dimensional data. As one of the most important layer types is the layer type that essentially performs *convolutions*, an image operation which the method derives its name from and that were already widely used in signal and image processing before CNNs became popular, we will first discuss this operation in Section A.3, before we discuss the layer types in Section A.4.

The process of training a CNN is then as follows. First, we initialise all the weights that we defined in our network, usually with a degree of randomness, but using heuristics to avoid bad weight initialisations. Namely, proper weight initialisation is an often overlooked factor for good convergence of the network. We then train the weights through iteratively passing our input data through the network and calculating the output, called a forward pass, after which we measure the difference with the ground truth according to a chosen error function, *backpropagating* this error through the network to calculate the extent to which each of the weights is responsible for that error in a backward pass, and finally adjusting the weights according to a chosen optimisation method. Completing a single forward pass and a single backward pass in this way is called an *iteration*. In current practice, the famous *backpropagation* algorithm is the most popular way to efficiently backpropagate the error to each of the individual weights, which we will therefore explain in detail in Section A.8. Within such an iteration, we can choose to process a subset of our training samples containing multiple samples at once, so that we update our weights using the values of all those samples simultaneously. This subset is called a *batch* and its size is called the *batch size*, which is an important hyperparameter. As we shall see, processing multiple samples has its advantages and its disadvantages. Each time we have completed such an iteration for each element in our training data set exactly once, we say we have completed one *epoch*. We commonly repeat this procedure until we have converged to a predefined desired minimum error value, or until we have completed a predefined number of epochs. For the optimisation method, by far the most common method is*gradient descent*, which, however, comes in many different varieties, as we will discuss in Section A.9.

Finally, in recent years CNN algorithms have been boosted significantly by the concept of *transfer learning*, which allows us to reuse models with pretrained weights and adjust them to our problem, as we will explain in Section 3.4.1, while we give an overview of the most popular existing models with available pretrained weights that we experimented with for our detection of intestinal content. In all of this, proper partitioning of our training and testing data remains as important as it was for learning through hand-crafted features. In general, however, we will need a significantly greater amount of data as we tend to have a far larger solution space, especially when training a CNN *from scratch*, i.e. without making use of any pretrained weights.

A.2 Architecture

With respect to the architecture, we will first explain the theory that general ANNs and CNNs have in common, as those concepts will be easier to grasp before we explain the specifics of CNNs. ANNs, essentially graphs in which its nodes, also called *neurons* in analogy to the brain, are grouped into *layers*, where neurons within a single layer are not interconnected. The idea of grouping of neurons into layers plays an important role in the functioning of CNNs as we will show below. Usually, there is at least one input layer, one output layer and one or more layers in between. These in-between layers are also denoted *hidden layers*, as only the input to the network and its final output are directly observed, while the inputs and outputs of the hidden layers are hidden from direct view. This does not mean, however, that their inputs and outputs cannot be observed if desired, as methods exist to visualise those. When referring to the number of layers, the common convention is not to count the input layer because the inputs are not active [71]. Thus, if we talk about a 2-layer network, we refer to a network with one hidden layer and one output layer. The neurons in a layer are then connected to neurons in one or more other layers through connections which are also called *synapses* in analogy to the brain. Each of these connections has a *weight* value assigned to it by which the inputs passed through that connection are scaled, such that the weights define the strength of the connection. Additionally, in every neuron, the input values are translated with a bias value. This is usually interpreted as an additional input to each layer with a value of 1, such that the weights of its connections to the next layer are in fact the bias values. In Figure A.1 we visualise a simple 3-layer ANN, where the bias values are indicated with b_i and the weights with w_i. As we can see, the bias values themselves are not very different from common weights when interpreted this way, as long as we consider that the input they are associated with is a constant.

Like the ANN architecture in our previous example, CNN architectures are often visualised with the layers ordered horizontally from left-to-right, where the rightmost layers represent the deeper layers. Alternatively, they can also be visualised with the layers stacked upon each other in a vertical manner, where the lower layers represent the deeper layers. Although both visualisations represent the exact same thing, for consistency we choose for a left-to-right approach in this work. Through these stacked layers, CNNs extract features while gradually reducing the output size of each layer, which are then fed to the classification layers that eventually output a probability for each class. For this, different layer types are utilised, each with its own purpose in achieving

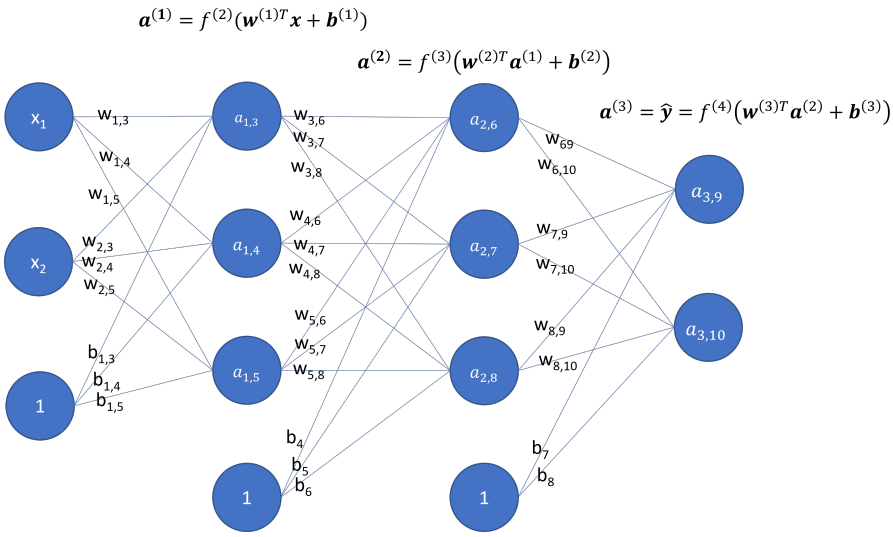


Figure A.1: An example of a 3-layer ANN architecture, input layer not counted as explained in the text, with weight values indicated by w_i and bias values indicated by b_i. The output of the network is the vector $\hat{\mathbf{y}}$ with 2 elements. The two middle layers are hidden layers, as their inputs and outputs are hidden from direct view.

the final classification targets, as we will discuss in Section A.4.

When a neuron receives the outputs from all of the connected neurons from the previous layer as its inputs, these are first individually scaled by the weights assigned to the corresponding connections. Then those values are summed, the bias is added and an *activation function* is applied element-wise to the result. Essentially, for each layer, the input is passed through the following equation:

$$F_l(\mathbf{x}, b) = f_a(\mathbf{W} \cdot \mathbf{x} + b), \tag{A.1}$$

where $\mathbf{W}$ is the matrix of weights, with in each row the weights corresponding to the connections of a single neuron, f_a is the layers' activation function, $\mathbf{x}$ is the vector of layer inputs and b is the bias term.

The main purpose of an activation function is to introduce non-linearities into the model that would be linear otherwise. Therefore it is desirable to choose non-linear activation functions at least in some layers, in which case they are often also simply referred to as *non-linearities*. We will address popular choices for activation functions and their pros and cons in Section A.6. Although the activation function is essentially a part of the neuron and could therefore differ between the different neurons of a layer, it is commonly defined as a

parameter of the layer, such that all neurons contained in that layer share the same activation function. Namely, this makes it possible to calculate the function over all neurons in a single matrix operation, allowing us to implement the forward pass of the network in an efficient dot product. Additionally, this paves the way for using activation functions that combine the values of different neurons, which is commonly the case in classification layers. We will therefore talk about the layers' activation function, rather than the neuron's.

In this way, CNNs can, inherently to general ANNs, essentially be seen as the definition of a set of functions that is parametrised by the weights of the network. It has even been shown that even when the network only contains one layer with a sufficient number of neurons with a continuous, bounded and non-constant activation function, this set of functions can approximate any continuous function [34]. Parameters thus play an important role in the definition of a CNN model, as, once a structure has been defined, they essentially determine the function it implements. The size or complexity of a CNN model is also commonly expressed in its number of parameters, as nearly all calculations in the CNN involve a parameter value, although there are cases where the number of neurons is used. Note, however, that not all parameters are weights or learnable parameters: a CNN training procedure is commonly also influenced by its hyperparameters.

Hyperparameters are all the parameters of our model that control or influence the learning procedure, but have to be set before we start training a network as they cannot be learnt through training. Here one should think not only of parameters that control the learning procedure, such as the learning rate, the optimisation algorithm, parameters to the optimisation algorithm (such as batch size or momentum in the case of the gradient descent, as we will see in Section A.9), and parameters of each layer that are specific to the layer type as we will discuss in Section A.4, but even of elements of the architecture itself, such as the number of neurons in each layer. In contrast, the learnable parameters are those parameters that we want to optimise through the training procedure. They essentially consist of the weights and the biases of all layers, although specific layer types may have additional learnable parameters.

The layers of a CNN are usually characterised by a common connectivity of the neurons contained within the layer to the neurons in the previous layer and/or other common parameters, such as the activation function. In this way, we can distinguish different types of layers. In the original ANNs, the most common layer type is the *fully-connected layer*, which has the property that all of its neurons are fully connected in a pair-wise manner to all of the neurons in adjacent layers. The main contribution of CNNs was exactly to replace this

layer with the focus on image input, as a fully-connected layer with a single neuron following the input layer for a CNN would already result in a large number of weights in the case of images. Namely, in the case of one of our CE image patches of 64×64 pixels with three colour channels, this would already result in $64 \times 64 \times 3 = 4096$ weights. It is easy to see how this number would increase quickly as we increase the number of neurons and layers, which is a likely desire for any non-trivial real-world problem. Not only does this quickly increase computation times during the training procedure of a network, but having a large number of parameters also makes the architecture prone to overfitting [85].

Overfitting is one of the major architecture-related issues with CNNs. The trade-off between the variety and the division of our input data, the depth of our network and the general architecture is important to consider. For example, the deeper the network is, the more complex the relationships are. Although this allows us to learn a separation between classes that adjusts more tightly to our input data, a slight variation between our training data and the domain we want our model to generalise to, can cause our model to overfit to the training data. Therefore, an important focus in CNNs is on several techniques that have been invented to prevent this and provide *regularisation*. This can happen both at the level of layer types and at the level of the the optimisation procedure and the cost function as we will discuss further ahead.

Apart from the issues in terms of computation, it can be argued that compared to a basic ANN, CNNs can learn relationships for images more efficiently performance-wise. Namely, we often seek for the appearance of a specific structure in the image which does not immediately depend on the remainder of the content of the image, making only the local neighbourhood of the object an area of interest to consider, such as a liquid bubble in our case of intestinal content. Instead of at a fixed location, this structure could appear anywhere and repeatedly in a single input image. Not only would using fully-connected layers for this result in an unnecessarily high quantity of weights, but at the same time multiple neurons would have to be responsible for recognising the exact same information where the only the order of the weights would differ, which it would have to learn independently. Therefore, CNNs implement the concept of *convolutional layers*, which are based on this exact insight that in each layer only a restricted neighbourhood of each input activation is relevant, while it is also sensible to share the parameters over different regions of the image. Despite often being implemented as a full dot product for calculation efficiency, where the convolution parameters, or simply weights, are duplicated,

it is equivalent to performing old-fashioned sliding window convolutions.

A.3 Convolution

The convolution operation has a long history in its application in image processing and signal processing in general. It is a form of filtering: an operation intended to remove unwanted components from a signal to emphasise the ones of interest. Specifically, convolution is a form of linear filtering, as it involves fixed-weighted combinations of input components, in our case pixels, in small neighbourhoods. Before we further expand on CNNs, we discuss this operation from the perspective of image processing as we believe it helps to create a better understanding the inner workings of CNNs and a foundation for making appropriate decisions on architectures and redefining models.

Convolution in image processing is a specific case of mathematical convolution. Mathematical convolution is defined as the integral of the product of two functions (or *signals*) with respect to an input value τ, denoted $f(\tau)$ and $g(\tau)$, after one of them is reversed and shifted by a value t. In the remainder of this section we take $f(\tau)$ to be our original function or signal, while we $g(\tau)$ denotes the function by which we modify our input signal to obtain a desired output signal. This integral is then calculated at all shift positions, which results in a new function over t, $(f * g)(t)$. Formally, the convolution can thus be defined as

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau. \tag{A.2}$$

The term convolution refers both to the operation and to the result function, as in "the convolution of $f(\tau)$ and $g(\tau)$". When t represents the time domain, we can consider the convolution as as weighted average of the function $f(\tau)$ at the moment t, where the weights are given by $g(-\tau)$ shifted by the amount t [81]. A nice property of the convolution operation is that it is commutative, i.e. $(f * g) = (g * f)$.

In the case of image processing, one of our functions is our image signal, which is essentially the function over the positions in the image, i.e. the x- and y-coordinates. The function we define to convolve our image with is called the *kernel function* or simply the kernel and we can simply imagine this kernel as a small image on its own. Usually, it is carefully designed by researchers with a certain objective, i.e. to filter the input image, to sharpen it, to blur it or to enhance certain structures. As defined in mathematical convolution, the kernel values are reversed and shifted, i.e. during the convolution the upper

left value of the area of our input image is multiplied by the bottom right value of the kernel. In this way, convolutions in image processing are used to detect patterns in the image signal by observing the output image, which is exactly the concept that is used in CNNs. The convolution operation sums the value of a pixel and those of its neighbours to the extent defined by the kernel size, weighted by the kernel values at the corresponding positions after reversing them, with the centre of the kernel of the filter aligned with the pixel in question. Formally we can define the discrete image convolution as follows:

$$(f * g)(t_x, t_y) = \sum_{x=-\lfloor \frac{w_g}{2} \rfloor}^{\lfloor \frac{w_g}{2} \rfloor} \sum_{y=-\lfloor \frac{h_g}{2} \rfloor}^{\lfloor \frac{h_g}{2} \rfloor} f(t_x - x, t_y - y) g(x, y), \tag{A.3}$$

where $f(x, y)$ is our image function, $g(x, y)$ our kernel function, h_g and w_g the width and height of our kernel respectively and t_x and t_y the position of our shift analogous to t in Equation A.2, or equivalently the location in the output image. For intuitive reasons, here we shifted and inverted our image signal $f(x, y)$, as this works better for imagining the operation as a convolution window that slides over the input image. We also gave the limits of the sums only for convenience, as the kernel function would not be defined outside of those limits and those terms would simply yield a multiplication by 0. Note that, due to the commutative property, an equivalent definition would be $\sum \sum f(x, y) g(t_x - x, t_y - y)$.

In practice, kernels that are used in image processing often tend to be symmetric which makes them identical to the the reversed versions of themselves. Notice that if we use an even-sized filter, there will not be a central pixel. In this case, the convention was originally is to anchor each input pixel to the right-hand value of the two central kernel values. However, as each time we then consider a greater part of the image to the left of the kernel than to the right, a non-symmetric filter shape would yield a non-symmetric filter response due to which the pixels in the output would be shifted, which would distort the image. In a CNN, where convolutions are applied sequentially as we will see, while being intervened by other operations, this shift could rapidly accumulate [97]. Therefore, even-sized kernels are rarely interesting in practice and are uncommonly seen in the context of CNNs and we will assume odd-sized kernels in the remainder of this section.

In the context of CNNs, it helps to clarify the nomenclature and convolution details. In this work, we will distinguish between filter, kernel and feature detector. Since the rise of CNNs, these terms have often been used interchange-

ably and become mixed up as CNNs and general image processing techniques grow further apart. The term *kernel* refers to a 2D-matrix, which defines the kernel function by which we *convolve* an input image. The entries of the kernel are somewhat confusingly called the *filter coefficients*. The term *filter*, however, refers to a concatenation of such kernels, which could be just a single one. In turn, feature detector refers to a general method for extracting features from an input image, and could consist of one or multiple filters and even other types of layers. Another slightly confusing aspect is that a CNN does not actually perform convolutions, but a related operation that is known as *correlation*. Namely, in this operation the signs of the offsets of the kernel function are not inverted:

$$(f \otimes g)(t_x, t_y) = \sum_{x=-\lfloor \frac{w_g}{2} \rfloor}^{\lfloor \frac{w_g}{2} \rfloor} \sum_{y=-\lfloor \frac{h_g}{2} \rfloor}^{\lfloor \frac{h_g}{2} \rfloor} f(t_x + x, t_y + y) g(x, y). \tag{A.4}$$

The correlation operation is also not commutative, contrary to the convolution. Although we will keep speaking of convolutions instead of correlations despite not reversing the kernel, as is common in CNN theory, it is important to be aware of this difference with traditional image filtering. In the CNN learning procedure, as we shall see, the filter values make up the vast majority of the weights that we optimise in the context of CNNs, thus aiming to find the optimal filters for feature extraction in the problem at hand without the need of manually designing them.

An important consideration in image convolution are the pixels at the border of an image. Namely, note that the input to $f(x, y)$ in Equation A.3 may attain values for which the function is not defined, i.e. there are no pixels at that position, due to which the function will cannot produce an output value at those locations. Although in image processing several solutions to this issue are popular depending on the case, in the context of CNNs only two of those solutions have been widely adopted. The first of those is zero-padding. In this idea, we use the value 0 in all cases where the original image function is not defined. This is equivalent to padding the image with as many zeroes as required for our specific convolution. Concretely, resulting from Equation A.3, we would pad the image with $\frac{w_g - 1}{2}$ zero values horizontally and with $\frac{w_h - 1}{2}$ zero values vertically. For example, in the case of a convolution with a 5×5 kernel, we would pad the image with two zero values in both directions. This solution ensures that our output image has the same dimensions as our original input image. However, it could introduce distortions along the edges, as we add invalid information to our original input image. Therefore, the second solution

is simply to ignore the boundary pixels and only perform the convolution at those positions at which $f(x, y)$ is defined in all cases, and thus only maintain the valid part of the convolution. With this solution, the dimensions of output image are decreased by the same value as the number of zeroes required in zero-padding, thus leading to an increased loss of information. Effectively, we can see that we lose the information at the border more rapidly than elsewhere in the image, as the border pixels now participate in increasingly fewer convolutions than the centre pixels participate in as they are nearer to the border. Still, in the case of CNNs there is a clear preference in literature for zero-padding, which we will discuss in Section A.4.

Visually, we can imagine a convolution as sliding our filter over the input image and can thus view it as a *sliding window* operation as we discussed in chapter 3.2.1. In this view, the filter is slid over all the pixels of the input image to calculate the convolution, which is often also called a *feature map*, especially in the context of CNNs, as it essentially shows where certain features can be found in an image. Remember from the previous section that in the case of CNNs, we also commonly apply an activation function before we obtain the final feature map. Equivalently, as it thus corresponds to the activations in the different parts of the input, it is also called an *activation map*. For example, consider the CE image shown in Figure A.3a. The values of its pixels are given in Figure A.2, where we also display the kernel values and the values of the feature map. The filter we use here is one that we may use if we wish to highlight diagonal edges. This filter is a good choice for the detection of lines as it has a response different from zero wherever the pixels along the diagonal differ significantly from the surrounding pixels on either side. Note that if this is not the case, the resulting convolution will commonly be closer to zero than at the edges, as the sum of all weights in the filter is equal to zero. The same convolution showing the corresponding images, i.e. the input image, kernel image and feature map, is given in Figure A.3.

Finally, we explain an idea from image processing that may serve as a bridge between convolutions and CNNs, namely the idea of *filter banks*. Instead of filtering an image using a sequential concatenation of convolutions to generate a single output, we can use several in parallel, thus having several parallel filters with each of them being applied to the image independently and generating a distinct output. Instead of only detecting edges into one direction, for example, we could simultaneously have a parallel filter that is responsible for detecting edges in the other direction. If we defined an entire set or bank of such filters that complement each other, we could detect different structures and combinations of them at once by feeding them to a classifier or combining

113	109	94	57	51	56	61	62	61	63
118	116	111	98	54	53	58	56	59	62
132	127	109	109	65	59	53	58	56	58
136	132	115	116	90	77	53	65	61	61
136	133	128	123	98	88	54	67	69	67
147	137	125	113	101	100	53	60	61	59
152	132	126	112	115	113	67	50	65	63
148	116	135	120	115	107	90	50	63	62
143	118	131	130	115	107	88	81	47	48
147	145	124	141	125	99	114	88	56	47
150	139	136	141	127	119	125	112	84	60

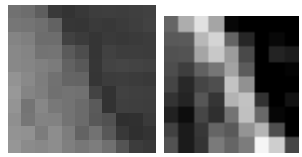
*

0	1	2
-1	0	1
-2	-1	0

=

101	173	225	165	7	0	0	0
89	90	184	212	100	38	0	8
81	83	165	200	156	73	0	35
77	62	97	118	163	91	0	0
69	40	69	78	195	145	0	0
51	36	86	53	179	194	50	0
33	13	70	47	115	192	104	6
65	27	55	101	89	148	181	78
82	27	52	122	101	124	245	206

Figure A.2: An example convolution to detect a diagonal edge over our input image. Note that the resulting convolution has decreased dimensions since here we used the valid area of the convolution instead of zero-padding around the edges. For clarity, we highlighted the values in the resulting image in the same colour as the values of the original image that were involved in its calculation. Note that the resulting value is given by summing all the values of the the multiplication of the values in the image area by the corresponding values of the kernel, after both rows and the columns of the kernel are flipped, which in this case leads to an identical arrangement.



(a) **(b)**

Figure A.3: The input image to the diagonal edge detection filter is visualised on the left-hand side, while the result is visualised on the right-hand side, both scaled by 4. Note that the output has fewer pixels, as the filter cannot be centred along the edges where there are not enough pixels on at least one side of the pixel.

or analysing the information in a different manner.

Several types of filter banks studied in literature have become generally popular for particular detection tasks. Some of those, such as the Leung and Malik (LM) filter bank, define a finite set of filters that can be used directly. Others, such as Gabor filter banks, define a wider family of filter banks originating from a generic filter definition, in this case the Gabor filter. Filter banks of the latter type were also used in work on detection of intestinal content detection in endoscopy images [91, 92].

This traditional image processing method, that has often been used for feature extraction in traditional machine learning methods, shares similarities with today's CNNs in the sense that CNNs also apply filters and combine the results. In fact, a filter bank is very similar to a single convolutional layer in CNNs as also suggested by Andrearczyk and Whelan [1]. Namely, we could interpret such a layer as representing an entire family of filter banks, with the difference that we now learn the values or weights of those filters through optimisation. However, CNNs further expand on this concept by concatenating several such layers, with each one thus defining its own family of filter banks, extracting each time more complex and more abstract features to comprehend in each deeper layer. Additionally, CNNs do not only consist of such layers, but are additionally intervened with non-linearities, pooling and other operations that further aid improved feature extraction and the learning procedure in general, as we explain in the following section.

A.4 Layer Types

We can roughly divide CNNs into two parts; the first set of layers that together perform the job of feature extraction, which we hereafter referred to as base-model, and the deeper set that together serve as classifier, hereafter referred to as top-model. The features extracted in the base-model are mainly extracted through convolutions in convolutional layers. While the first layers of the network extract simple low-level features, through concatenating more layers, a CNN can learn to detect features of higher complexity, each time through non-linear combinations of features from the previous layers [2]. Such non-linearities are introduced by defining activation functions that transform the outputs of each convolutional layer in a non-linear manner, as we will discuss in detail in Section A.6, thus introducing converting an otherwise linear model.

Apart from the convolutional layers, there are important other common layer types that aid the feature extraction in the base-model in distinct ways. Often,

they do so by reducing dimensionality and introducing further non-linear relationships between features, which all together help to prevent overfitting to the training samples and improve the generalization performance of the network. Although researchers may define any sort of custom layer specific to a certain task, there are standard layer types that have become widely adopted in practically all fields where CNNs are applied which we will discuss here, namely convolutional layers, pooling layers, batch normalization layers, drop-out layers and softmax layers. Most of these layers have important hyperparameters, other than the number of neurons within that applies to all of them, which are specific to the layer type. Here we discuss each of these layer types in detail, along with their most important functions and hyperparameters.

A.4.1 Convolutional

The convolutional layer is the core of a CNN, responsible for extracting features from our input images that become more detailed and complex as we concatenate them, intervened with well-chosen non-linear activation functions other layer types. However, the extent to which we obtain richer features when concatenating more layers always depends on our input data; both on the amount of data and the variety we capture within. In a convolutional layer, the weights that we aim to learn are in fact the values of the kernel or convolution window, as described in Section A.3. Although the operation of the convolutional layer is equivalent to the traditional convolution, it is implemented through a full dot-product with weight sharing between the different inputs of a filter. We explain the weight sharing scheme further ahead in Section A.8. As shown in Equation A.1 for the general case, in convolutional layers there is also a bias term involved. In this case, the bias term is shared for a filter just as the weights are, through the same sharing scheme.

First and foremost, it is important to understand the term *filter* in the context of a CNN. As discussed in Section A.3, this term is often used interchangeably with kernel or feature extractor. We already explained the general nomenclature in that section. However, in the context of a CNN, *filter* more concretely refers to the stack of kernels of depth C, with C being the number of channels in the layer input, which together output a single feature map. In case the convolutional layer is the first layer of the network, this number is commonly 3, as our inputs are usually RGB images as in the case of our work. Deeper in the network, this number tends to be higher as we tend to use more than three filters per layer, thus yielding an output of $C > 3$.

In Figure A.4 we show how a convolutional layer operates, displaying the case of a single filter. As we can see, a filter has the same number of channels as the input with different weights for each channel, where each channel c_i operates on input channel c_i. The outputs over the different channels are then summed to create a single output value for each point of the input feature map over the entire depth of the input. In this figure we can also observe the effect of some important hyperparameters, namely the stride S and kernel size $w \times h$, which we will explain in this section along with the other main hyperparameters, padding and weight initialisation. The stride, kernel size and padding are highly correlated, as different combinations of values for these hyperparameters can render the network structure invalid, or invalid for images of a certain input size. It is also important here to mention the term *receptive field*. With the term receptive field, people often refer to the pixels of the original input image that are involved in the calculation of a single value in the output feature map. Thus, as we progress through the network in a forward pass, the receptive field size continuously increases as with each convolutional layer, and, as we shall see, with each pooling layer, we incorporate information that originates from more distinct pixels in the input.

First, the stride determines the connectivity to the previous layer. It refers to the distance between successive convolutions performed by the same filter. We can set different values for the stride in the horizontal and vertical directions. Next, the kernel size determines the size of our convolution window and is commonly chosen to be an odd value for reasons that should be clear from Section A.3, where the height and the width can differ from each other if desired to create a non-square filter shape. Finally, padding allows us to deal with the border values in different ways, which we already discussed in Section A.3. The most common option is zero-padding, while the next most common option is to retain only the valid part of the convolution. The reason zero-padding is the more popular choice, is related to the dependency on the other parameters that we briefly mentioned. Namely, when using zero-padding, we can think of different ways of padding the image. For example, we could pad the image with two zero values on the left of the image and only one on the right. However, in the vast majority of cases we will simply want to add as many zeroes as required to obtain output dimensions of $\frac{1}{S}$ times the original to have dimensional consistency. Therefore, popular CNN software libraries allow the user simply to specify the usage of zero-padding, after which it determines the required amount of zero-padding automatically to yield an output of the same spacial dimensions as the input, depending on the input size and the other parameters. Guaranteeing dimensional consistency as our input is passed through the network, allows us to reason intuitively

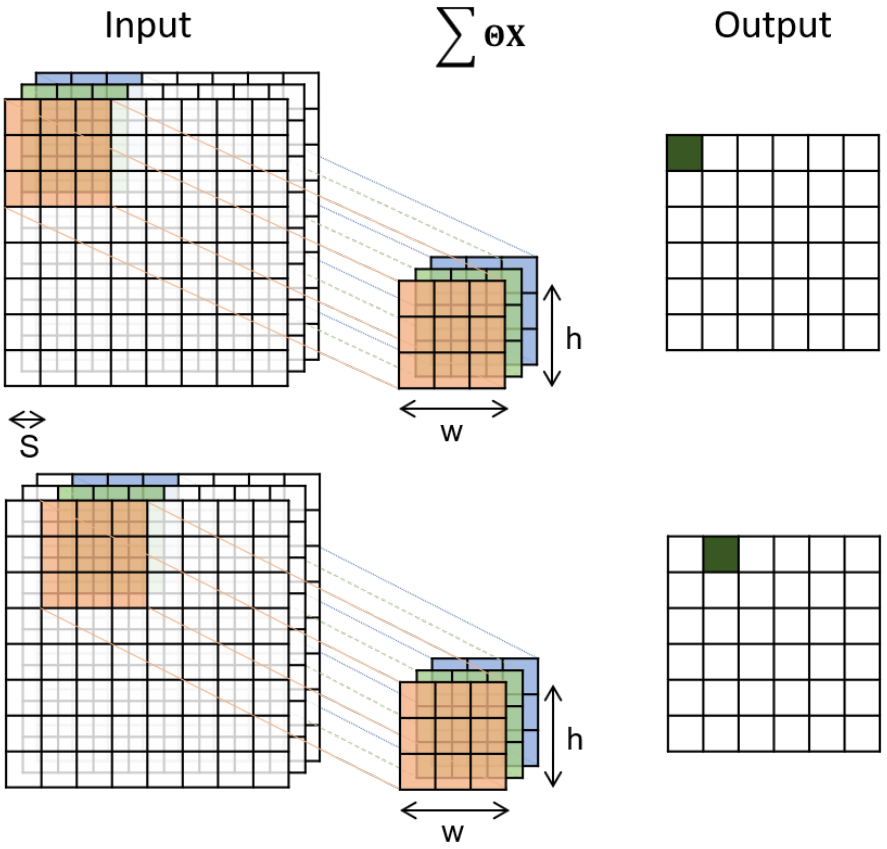


Figure A.4: An example of the calculation of two adjacent output values through the convolutional layer, with an input of $8 \times 8 \times 3$ and an output of $6 \times 6 \times 1$. Note that to compute the value marked in dark green in the output, all of the values that are marked in the input feature map involved, each with its own specific weight as specified in the filter. As the filter weights are shared for all inputs, the filter values in the calculation of the second output value, displayed in the bottom operation, are exactly the same as those in the upper part. Relevant hyperparameters are indicated: a stride of 1 and a filter width and filter height of 3 each.

about the size reduction of the feature maps throughout the network, as the width and height output dimensions of the layer then thus only depend on the stride parameter. As with traditional convolutions, it is often argued that using zero-padding allows us to take the information around the borders of the image further throughout the network, though it has to be noted that this information is polluted with non-information we introduce in the form of padded values. If we instead decided to use the valid part of the convolution only, then even with a stride of only 1 our dimensions would gradually be reduced as we concatenate convolutional layers. We can also see how in this case the loss of the information at the border rapidly accumulates, while we would carefully have to keep track of the input dimensions and the output dimensions of each layer to ensure that our architecture is valid and converges properly. Using zero-padding thus avoids extra complications in the design of the architecture and simplifies its reusability for new input sizes. Additionally, it is often argued that zero-padding allows us to create deeper networks, as the size of the feature maps is not reduced as quickly. A final important reason that the choice for the valid part of the convolution only is not as popular, is that this choice would prevent us from using shortcut connections, i.e. connections between a neuron and a layer further ahead, skipping at least one layer in between, as the dimensions would no longer match. Many widely adopted architectures, or modules introduced by such architectures, implement such connections, as we will see in Section 3.4.1.

The last hyperparameter we discuss here, is the initialisation of the weights of our network, which can be based on different theories. If we start learning from scratch, we most commonly want to initialise our network with random weights, but heuristics have been investigated to improve the initial weights by drawing values from certain distributions that have been investigated in literature for this purpose. For special cases, we may opt for different types of initialisation, such as in transfer learning. Weight initialisation methods are further discussed in Section A.8.

Formally, we can now define the convolutional operation as for a single output value in our output feature map as follows:

$$f(i,j,o,\mathbf{X}) = \sigma\left(\sum_{x=-\lfloor w/2\rfloor}^{\lfloor w/2\rfloor}\sum_{y=-\lfloor h/2\rfloor}^{\lfloor h/2\rfloor}\sum_{u=1}^{N}\Theta_{x,y,u,o}\mathbf{X}_{si+x,sj+y,u}\right), \tag{A.5}$$

where i, j and o are the dimensions in the width, height and depth of the the layer output feature map respectively, $\mathbf{X}$ is the layer input matrix, w and h are respectively the width and the height of our filter, s is the layer's stride

parameter and N is the number of channels in the layer input.

A.4.2 Pooling

Pooling layers are layers that apply a function over a small area of adjacent pixels in order to produce a single output value with the objective of progressively reducing the spatial size of the feature maps, which reduces the number of parameters in subsequent layers along with the computation time. At the same time, their input connections have no weights, so that they introduce no extra parameters and barely any overhead to our CNN model. The function that is most often used for this purpose is the *max*-function, although historically other popular choices are the average function or the L_2-norm. It has two hyperparameters, which are the stride and the size of the pooling area. In the state-of-the-art, most often the size of the pooling area is chosen to be 2, along with a stride of either 2 or 3, as larger pooling areas generally quickly leave out too important information. Formally, we can summarise the pooling operation as

$$f(i,j,u,\mathbf{X}) = \left(\sum_{x=-\lfloor w/2 \rfloor}^{\lfloor w/2 \rfloor} \sum_{y=-\lfloor h/2 \rfloor}^{\lfloor h/2 \rfloor} |\mathbf{X}_{si+x,sj+y,u}|^p \right)^{1/p}, \tag{A.6}$$

where i, j and u are the dimensions in the width, height and depth of the the layer output feature map respectively, $\mathbf{X}$ is the 3-dimensional feature map that is the layer input, w and h are respectively the width and the height of our pooling area, s is the layer's stride parameter and p is the p-norm value. If we set $p = \infty$, we obtain L_∞, which is commonly known as the *max*-function.

Lately, it has been argued that in many architectures pooling layers may be unnecessary as convolutional layers with equal stride and filter size would outperform such pooling layers in a fully convolutional neural network [82], noting that the number of parameters would increase. The authors of that work argue that a pooling layer can in fact be seen as a special case of a convolutional layer with equal stride and filter size, but with a p-norm activation function and a weight-matrix fixed in such a way that the operation is applied to each channel of the feature map independently. We can derive the case of pooling from Equation A.5 if we fix the weights such that $\Theta_{x,y,u,o} = 1$ if $u = o$ and 0 otherwise. Through experiments, they provide support for their hypothesis that among the three aspects that differ between common convolutional layers of stride and pooling layers, i.e. the p-norm activation function and the

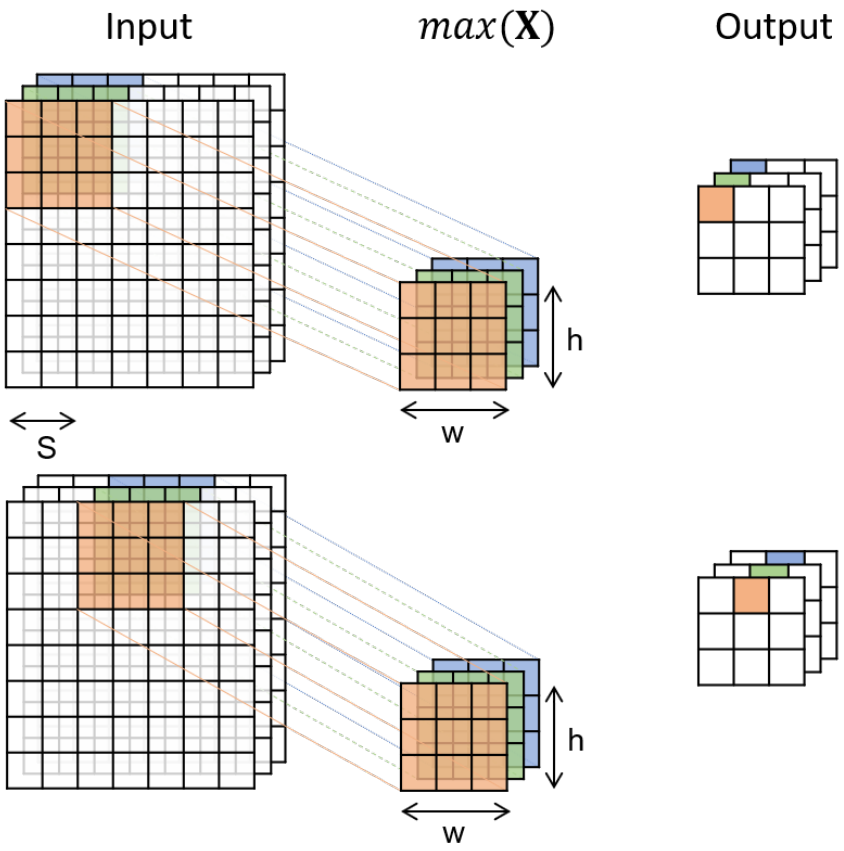


Figure A.5: An example of the calculation of two adjacent output values through the pooling layer, with an input of $7 \times 7 \times 3$ and an output of $3 \times 3 \times 3$. Here, contrary to the convolutional layer in Figure A.4, each depth channel has its own output value, i.e. the values marked in different colours are not combined. Hence, the pooling operation is only performed over each 9 values of the same colour to yield the corresponding value in the output feature map. This time, we use a stride of 2 as we are pooling, with a filter width and filter height of 3 each, thus having a degree of overlap between pooling operations.

spatial dimensionality reduction that makes the network capable of covering larger parts of the input in deeper layers, only the latter is crucial for achieving good performance. In similar experiments they show how equivalent convolutional layers can replace pooling layers with improved performance. As per the terminology the authors introduce in this work, architectures that replace pooling layers in this way are often denoted fully-convolutional networks. We were interested in this concept in our work, and experimented with replacing pooling layers by convolutional layers. In fact, we can see that the architecture we eventually came up with in Figure 3.16 is fully convolutional.

A.4.3 Drop-out

Drop-out layers are another form of input regularisation.The purpose of drop-out layers is to prevent overfitting to the training samples. It does so by randomly disabling a node of the previous layer at each step during training time with a frequency that is a hyperparameter of this layer type. Commonly, drop-out values between 0.2 and 0.5, thus dropping 20% to 50% of the input nodes respectively each time, are recommended in function of the size of the network. Namely, we would have to assure that not too many nodes are being dropped, so that the remaining nodes in the architecture are can together still be representative enough for our input samples. Drop-out is generally less effective for convolutional layers than for fully-connected layers [87], which is why it is more recommended for regularising the top-model. Namely, convolutional layers have spatial relationships encoded in feature maps due to which activations can become significantly correlated, in which case randomly dropping nodes would do more harm than good to the learning procedure. It has been argued that often, drop-out layers can be replaced by batch normalisation layers for their regularising effect [36], while batch normalisation has other benefits than to prevent overfitting. We also experimented with this in our intestinal content classification architecture.

A.4.4 Batch Normalisation

Batch normalization layers normalise the output of the previous layer [36]. They do so by calculating the mean and standard deviation for its input batch, after which the batch mean is subtracted from each of the inputs and the result is divided by the standard deviation of the batch: $F_l(\mathbf{x}) = \frac{\mathbf{x}-\mu_b}{\sigma_b}$. This layer thus basically applies one of the most common forms of feature normalisation from traditional machine learning techniques, albeit now applied

over the batches that we feed as the input to the CNN instead of all of the training samples. Batch normalization layers have a regularising effect, as in each step different examples are randomly included in the batch over which the output is normalised, due to which we can consider the batch mean and standard deviation as values with a degree of randomness. As both of them are used to modify the neurons' output values, this concept introduces variation in the layers' outputs, which, in turn, means that the layer has to be more robust to deal with this variation. At the same time, by ensuring zero mean and unit variance throughout the training procedure, decreasing the relative differences between different input values once scaled and translated by the weights and bias terms, it helps to fight the vanishing gradients problem. We will revisit this problem when we explain in Section A.8, where we also explain the importance of gradients in the optimisation of CNNs in general.

A.4.5 Softmax

Even though Softmax is not actually a layer type, but an activation function, we still list it here as it is the most common activation function in the output layer in classification CNNs, which is therefore often referred to as the Softmax layer. In fact, it is simply a fully-connected layer, of which the number of neurons corresponds to the number of output classes, so that it assigns a score to each of them, with Softmax as its activation function. As we will discuss in Section A.6, the choice of the Softmax activation function ensures that the values of this layers' activations sum to one, and, in turn, allows for the interpretation of those values as probabilities. This activation function is different from most others as it does not take a single value as input to produce a single output for a neuron in the layer, but instead considers all neurons at once, requiring the sum of their input values in its calculation.

A.5 Loss Functions

Loss functions or cost functions are a core component of the training procedure that measure our disagreement with the output of the CNN, or that, in other words, quantify the difference between the output and our ground truth labels. Although researchers can define their own loss functions based on their specific problem, there are a few popular loss functions that appear to work well in general cases or with specific popular architectures. Below, we list and briefly

discuss the most important ones of these. Note that we focus on loss functions for classification networks, not for regression in general.

A.5.1 Cross-Entropy

The cross-entropy loss function is one of the most used loss functions in CNN classification and is often considered to be a generalisation of logistic regression to multiple classes. The loss function used in logistic regression is the negative log-likelihood, commonly also called *log loss*, and is popular in maximum likelihood estimation in statistics. Strictly speaking, these are in fact different functions with different origins, but calculate the same value under the conditions of CNN classification problems, due to which the terms are often used interchangeably. We will not dive further into this as it is not necessary in the context of CNNs, but we refer interested readers to the excellent explanation by Goodfellow et al. [29]. We do note, however, that from a statistic points of view the cross-entropy function measures the *entropy* between two probability distribution. The entropy refers to the average bits of information, or in other words the average amount of "surprise" a probability distribution represents. Balanced probability distributions have a high entropy, while probability distributions with a very high probability for a specific event have a relatively low entropy. The cross-entropy then measures the entropy of the combination of two distributions, which essentially sums up to the difference in entropy plus the entropy of the original distribution. In fact, it is much related to the *Kullback-Leibler (KL)* loss function, which simply corresponds to the difference between the cross-entropy and the entropy of the original distribution.

Often, theory and software packages distinguish between binary cross-entropy and generic or *categorical* cross-entropy, while the binary cross-entropy is actually a special case of the categorical function that cross-entropy loss function. The binary cross-entropy function applies to a classification problem between two classes, here referred to by class 1 and class 2. As this function is the most intuitive to understand, we will first discuss that definition to then deduce the generic function. Binary cross-entropy is defined as:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -y_1 \log(\hat{y}_1) - (1 - y_1) \log(1 - \hat{y}_1), \tag{A.7}$$

where $y_1 = 1$ if the ground truth label of the sample corresponds to class 1 and $y_1 = 0$ if it corresponds class 2, while p_1 is the predicted score for class 1 by the algorithm. This formula shows why it is also called log loss, as we take the logarithm of the predicted values. If the sample thus corresponds to class

1, only the first term remains and the formula is reduced to $-\log(p_1)$, while in the other case, only the second term remains and the formula is reduced to $-\log(1-p_1)$. We can thus observe that the function can always be reduced to $-log(p_c)$, in which p_c is the predicted score for the class the sample corresponds to, i.e. $y_c = 1$. We thus attempt to minimise the negative logarithm of p_c, i.e. the score assigned to the correct class. The closer this value is to 1, the correct prediction, the smaller this term is, as the logarithm of values between 0 and 1 is negative. From this observation, we can finally deduce the categorical cross-entropy function for multiple, mutually exclusive classes

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{c=1}^{M} y_c \log(\hat{y}_c), \tag{A.8}$$

where M is the number of possible output classes, y_c is 1 if the sample belongs to class c and 0 otherwise, and p_c is the predicted scores for class c by the algorithm. It is relevant to note that if the classes were not mutually exclusive, i.e. a sample can belong to multiple classes at the same time, also called multi-label classification, we would instead use the binary cross-entropy function multiple times.

The cross-entropy function has the interesting property that its derivative with respect to the inputs of the (softmax) activation function, also called *logits*, as used in backpropagation, is trivial to calculate. Namely, this derivative is simply $\mathbf{p} - \mathbf{y}$.

A.5.2 Kullback-Leibler Divergence

As mentioned above, this loss function is much like the cross-entropy function, but instead measures only that part of the entropy that is due to the difference between the probability distributions. The result of this function plus the entropy of the original distribution, together make up the entire cross-entropy loss. This is formally defined as

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{c=1}^{M} y_c \log\left(\frac{y_c}{\hat{y}_c}\right). \tag{A.9}$$

A.5.3 Hinge Loss

Hinge Loss is the maximum margin loss function that is also used in SVM classifiers. This classifier is only intended for binary classification, although extensions exist for multi-class classification that are not widely used and which we did therefore not consider in this work. Using this classifier in fact turns the classifying layers of the CNN into an equivalent of an SVM classifier, as it attempts a maximum margin between positive and negative samples. Its binary definition is

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \max(0, 1 - \mathbf{y}^T \hat{\mathbf{y}}), \tag{A.10}$$

where $\mathbf{y} \in \{-1, 1\}$ is our vector of ground truth labels and $\hat{\mathbf{y}}$ the vector of scores assigned by the model. Note that, in contrast to the other loss functions we described, this loss function thus requires us to modify our target variables from the labels, commonly 0 and 1, to -1 and 1 and respectively. We also have to be careful in our choice for the activation function of the output layer, as our outputs should at least cover the range of -1 to 1, which means sigmoid cannot be used directly and tanh, for example, may be a more appropriate choice.

Also note that, effectively, the minimisation of this loss will only consider examples that infringe the margin, as for all other samples the function would yield 0 and thus a 0 gradient, comparable to the support vectors in SVM classification. Although the function itself is not differentiable, the gradient is easily computed locally.

A.5.4 L_1-loss and L_2-loss

Although the L_1 and L_2 loss functions are used in regression problems and are in fact not used stand-alone in classification problems, they are still useful to consider here, as they are often added to other loss functions as regularising terms. When used stand-alone, they thus apply when the output values represent metrics or scalars of a wider variety.

L_1-loss, also referred to as mean absolute error (MAE), is based on the L_1-norm, which, in turn, is alternatively referred to as least absolute error (LAE) or least absolute deviations (LAD). It is defined as

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = ||\hat{\mathbf{y}} - \mathbf{y}||_1 = \sum_{i=1}^{i=N} |\hat{y}_i - y_i|, \tag{A.11}$$

where $\mathbf{y}$ is the vector of outputs for our samples, $\hat{\mathbf{y}}$ the vector of ground truth labels and N the number of samples. It thus yields the sum of all the individual distances of our samples to the ground truth.

Alternatively, L_2-loss is used, which is commonly known as the mean squared error (MSE), least squared error (LSE), Euclidean loss or simply sum of squares, although the terms are technically not equal, and is based on the L_2-norm. Instead of minimising the absolute differences, we now minimise the squared differences:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = ||\hat{\mathbf{y}} - \mathbf{y}||_2^2 = \sum_{i=1}^{i=N} (\hat{y_i} - y_i)^2, \tag{A.12}$$

where $\mathbf{y}$ is the vector of outputs for our samples, $\hat{\mathbf{y}}$ the vector of ground truth labels and N is the number of samples. As Equation A.12 shows, this loss function does in fact not compute the L_2-norm, but the squared L_2-norm. Although both the L_2-norm and the squared version would provide the same optimisation objective, the squared L_2-norm is often preferred as it is computationally more efficient as it avoids the square root.

The most interesting aspect of L_1 and L_2 loss in this work, however, is their use as regularisation terms by adding them to any other loss function we may use. Here, instead of applying the L_1-norm and L_2-norm to the differences between predictions and ground truth labels, they are simply applied to the values of the weights and multiplied by a parameter λ that determines the amount of regularisation. In this sense they help to keep the values of the weight matrices low, which is understood to lead to simpler models and reduce overfitting to a certain extent.

In general, L_1-loss is more robust in the presence of outliers, but as its derivatives are not continuous it is inefficient and unstable in optimisation. In contrast, L_2-loss is sensitive to outliers, but is more stable and more efficient [98]. Therefore, unless the dataset contains outliers that are important to take into consideration, L_2-loss often preferred. If required, preprocessing methods can be used to remove outliers beforehand. While as regularisation terms both L_1 and L_2 are used to keep the weights small and therefore the model less complex, L_1 has the property to force weights all the way to zero, while L_2 ensures they never actually get down to zero merely due to regularisation. As having weights of zero is often undesirable as we will explain in Section A.8, the L_2-norm is relatively more popular for this purpose.

A.6 Activation Functions

Sometimes specified as a separate layer, activation functions are vital to the learning process of CNNs. The concept of an activation function derives its name from the fact that this function determines whether the neuron *activates*, i.e. whether it produces a signal at all as well as the magnitude of this activation. For a neural network to learn non-trivial relationships, it is necessary that such functions are non-linear to reproduce a non-linear function as briefly mentioned above. If only linear functions were chosen in several concatenated convolutional layers, we would simply be able to reduce all of those layers to a single one by concatenating all the weight matrices. Therefore, here we will discuss several non-linear activation functions that have become popular in CNN theory, or were so historically.

A.6.1 Logistic function

The logistic function is also often referred to as "the Sigmoid function", even though a sigmoid function is any mathematical function that has a characteristic S-shaped curve and as we will see, the Tan-h loss function is also a sigmoid function. The logistic function is defined as follows:

$$f_a(x) = \frac{1}{1 + e^{-x}}$$

It was was suggested and frequently used historically in basic neural networks, because it has the attractive property of squashing real numbers to range between 0 and 1. Specifically, large positive numbers are clipped to 1, while large negative numbers are clipped to 0, which historically had the interpretation of a neuron being maximally activated, i.e. emitting a signal at an assumed maximum frequency, or being inactivated, emitting no signal at all. However, due to this it has the tendency to kill gradients when the activation of the neuron is saturated at either end of the tail of the sigmoid. Another problem is that the output of the logistic function is not zero-centered, which would make the gradients during back-propagation either all positive or all negative. This is only an inconvenience though, as processing the data in batches would cause the updates to the network to have variable signs.

A.6.2 Tan-h

The Tan-h loss function, or hyperbolic tangent, is also sigmoidal, but has the advantage that it is zero-centred as it symmetrically squashes the output between -1 and 1 instead. It is in fact a shifted and scaled version of the logistic function:

$$f_a(x) = \tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

Because it is zero-centred, it is often preferred to the logistic function, but it still suffers from saturation.

A.6.3 ReLU

As a solution for both of these problems, as well as being significantly more efficient, the ReLU function was suggested and has now been widely adopted in CNN architectures. The ReLU function simply clips any negative values to 0, while positive values remain untouched:

$$f_a(x) = \max(0, x).$$

Despite its simplicity, it has been shown to accelerate convergence of stochastic gradient descent and is inexpensive to compute, especially compared to the aforementioned non-linearities. The ReLU function is also not completely free of issues though. Namely, ReLU activations can become unreversibly inactivated during training if a large gradient causes the weights to be updated in a certain way, which can become especially frequent with high learning rates. With an appropriate configuration of the learning rate, this is less likely to be an issue.

A.6.4 Leaky ReLU

There are some variations on the ReLU function which mainly attempt to generalise concepts of ReLU or solve the issue of dying neurons, such as Leaky ReLU. Leaky ReLU instead has a small negative slope, i.e. αx in which α is a very small constant to ensure that the output will never be equal to 0:

$$f_a(x, \alpha) = \max(\alpha x, x).$$

A.6.5 ELU

The exponential linear unit (ELU) is another modification of the ReLU function. For input values equal to or higher than 0 it follows the same slope as ReLU, but for input values lower than 0 it has an exponential slope:

$$f_a(x, \alpha) = \begin{cases} x, & \text{for } x \geq 0 \\ \alpha(e^x - 1), & \text{for } x < 0 \end{cases}$$

A.6.6 Maxout

Finally, a different type of neuron has been proposed, Maxout, where the activation function is not simply a function of the single input value of the dot product of the weights and the data summed by the bias, but where there is a second input value defined by a second set of weights and a second bias. Maxout is a generalisation of both ReLU and its leaky function, allowing for the configuration of both the slope and the offset of the two different slopes through optionally learnt parameters:

$$f_a(x, \alpha, \beta, b_1, b_2) = \max(\alpha x + b_1, \beta x + b_2).$$

Although optimising the parameters through the network training procedure may sound attractive, it doubles the number of parameters, while the problems with ReLU can relatively easily been overcome through an appropriate configuration of the learning rate. The ReLU function is therefore considered a good starting point in the design of new CNN architectures when there is no clear preference for another specific function, while researches can later experiment with other activation functions if results are not satisfactory. In our work on intestinal content detection we experimented with the use of Sigmoid, Rectified Linear Unit (ReLU), Exponential Linear Unit (ELU) and Leaky ReLU.

A.7 Weights Initialisation

CNNs often make use of stochastic algorithms for the optimisation in the training procedure. The best way to view this is by interpreting the algorithms as search algorithms, that search for the optimal solution in a landscape of many different combinations. If we imagine a visualisation of this solution space, we may do so as a manifold where every point on the manifold represents

a different combination of parameters. Our global solution would then be the lowest point at the manifold, while there are many locations where the manifold is locally low. Deterministic algorithms would explore the whole solution space and we can be certain they find the global optimum and make guarantees about its running time. However, with this manifold growing in vastness and complexity due to an increased number of outcomes or size of the input data, the problem may become too hard to be solved by deterministic algorithms in reasonable time, leading us to opt for stochastic algorithms. These are algorithms that use a degree of randomness or uncertainty to solve the problem and therefore usually find a solution that is not the global optimum, but a local one. They are non-deterministic algorithms, although a more reliable class among those, as they quantify the solution space in terms of probabilities instead of mere possibilities [75]. This allows us to estimate the probability that we find a solution that is as good as the global optimum for our purpose.

The use of stochastic algorithms in CNNs implies that, as opposed to deterministic algorithms, a different, likely non-optimal result is produced each time if the algorithm is run multiple times. This means that each time we train a CNN model, the model we produce will be slightly different, with different evaluation results. Concretely, in CNNs this randomness commonly at least applies to the initialisation of the weights, biases and to the division of the training data. The partitioning of the training data generally applies to machine learning problems. The randomness in this partitioning can be kept constant. For research purposes, we can choose to partition our data into training, validation and test data beforehand, in order to keep the data division constant and eliminate this factor of randomness. While we would still use a degree of randomness to partition the data, we could use this same partition to train different models or test different values for parameters. It is also desirable to use multiple such partitions, as we do in cross-validation, as we are then less likely to get stuck with a single partition of the data for which our model always convergences to a sub-optimal solution for the data domain in which our CNN should operate. In contrast to the data division, in the decision about initialising the weights we do not have as much information. In data division we can assess and decide on the variety, origin, e.g. originating from different patients, and conditions, e.g. different lighting, by visualisation or other forms of data analysis.

The solution space of the values of the weights in a CNN architecture is more difficult for us to interpret. Therefore, a degree of randomness in the initialisation of weights is always desirable. Once we have found good initial weights, we may choose to keep them constant in a production environment

or for research purposes, to allow us to measure the effects of changes in our data set or changes of the other parameters. Care has to be taken for the initialisation of the weights, as the final solution could be largely dependent on the initial weights. Poor weight initialisation, for example when the weights are relatively too close to zero, could even cause gradients to vanish in back-propagation; this is known as the vanishing gradients problem, which we will revisit in Section A.8. There are different ways to incorporate randomness into the initialisation of the weights. In essence, there are three popular options for drawing our random samples: we can draw them from a uniform distribution, a normal distribution and a truncated normal distribution. For these distributions, we can further specify the desired distribution parameters, i.e. the mean and standard deviation in case of the normal distribution and the mininum and maximum values in case of the uniform distribution. Apart from this method, there has been research on the use of heuristics that can improve or speed up convergence of the training procedure that can increase the chances of finding a global optimum. Glorot and Bengio showed in their work how drawing the weights straight from certain random distributions could increase the probability of the occurrence of vanishing gradients and suggested heuristics to avoid this problem [28]. Concretely, they suggest to draw the weights from a truncated normal distribution with a mean of 0 and a standard deviation of $\sqrt{\frac{2}{n_{in}+n_{out}}}$, where n_{in} is the number of input connections and n_{out} is the number of output connections. Their method is now popularly named the "Xavier initialisation" or "Glorot initialisation", and is currently the default of the popular Keras (Tensorflow) software package. Other random initialisation options are using an orthogonal matrix obtained from a matrix of random numbers or initialisation from a random distribution with variance scaling, i.e. the variance of the random distribution from which the weights are drawn is scaled to a specified extent with the number of input or output units. The latter is essentially a generalisation of the Xavier initialisation. Other often used options that do not involve randomness and are only recommendable for specific circumstances are initialisation with a constant value (including 0 and 1) or with the identity matrix.

A.8 Back-propagation

Once the weights are initialised we can use those values in a forward pass of a *batch* of our inputs through the network, ultimately producing output values. In first instance, if the weights have not been loaded from a pretrained network with specific domain information, the output will likely differ significantly from

the desired output that is in our case given by labels or annotations of the images. The degree of this difference or error is given by the loss function we have chosen to use, as discussed in Section A.5. But how do we then use this function to actualise and improve the weights, and thus eventually optimise the network?

The only widely accepted manner to optimise CNNs for this nowadays is through *backpropagation*, which is an algorithm to efficiently calculate the derivatives of the error function with respect to each of the weights to optimise their values according to a chosen optimisation method [95]. The gradient in fact makes up an important part of the limited information we have about the surface of the solution space (alternatively named error surface in this context). As analysing the whole solution space in the context of a CNN is unnecessarily complex, the gradient of the loss function gives us information about the direction of the steepest slope of the error surface, which, in the absence of general information about the solution space, is highly useful information about the direction towards a minimum. To be able to use this information, of course the loss function has to be differentiable and we should be able to calculate the gradient efficiently.

In this sense, it is difficult to see backpropagation algorithm loose from the optimisation method, commonly gradient descent, which uses the calculated gradient values for steering the training procedure towards the direction of a minimum. It is, however, important to make this distinction. Namely, backpropagation strictly refers to the algorithm that efficiently computes the gradient of the loss function with respect to each weight, while making no assumptions about how the weights are updated afterwards. Instead of separately calculating this gradient for each of the weights independently, which would be highly inefficient and involve calculating intermediate values many times, the backpropagation algorithm does this through combining the layered structure of neural networks with intelligent use of the chain rule for derivatives, using dynamic programming. To clarify this, let us visualise backpropagation on a simplified situation of a basic ANN in Figure A.6. Refer to this image for the variables that we discuss below and to keep an overview of how we deduce the error back to the individual weights.

In this network, imagine we have activation function $f^{(l)}(x)$ in layer l, which could be the ReLU function as we discussed in Section A.6, or the Sigmoid function. Additionally, let the weights matrix of layer l be given by $\mathbf{W}^{(l)}$ and the input vector, as opposed to the matrix we would have in the case of a CNN,

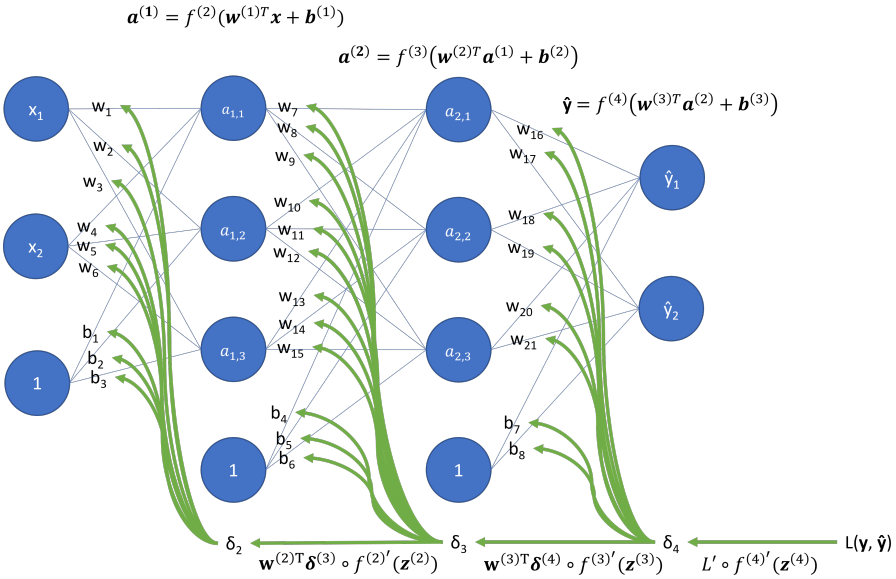


Figure A.6: An example of a three-layer ANN on which we visualise the backpropagation principle. Note that $L(\mathbf{y}, \hat{\mathbf{y}})$ is the loss function which measures the error between the prediction and the ground truth, while the δ_l values are the error or negative gradients with respect to the layer input values, scaled by their weights and translated by the bias values. These values are only calculated once, after which they are reused in the calculation of the gradients with respect to each weight individually, as we propagate the values further backwards through the network.

by $\mathbf{x}$. In that case, we can compose the function calculated by the network:

$$g(x) = f^{(3)}(\mathbf{W}^{(3)} f^{(2)}(\mathbf{W}^{(2)} f^{(1)}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) + \mathbf{b}^{(3)}). \tag{A.13}$$

However, backpropagation theory would get unnecessarily complicated if we did not define additional helpful variables. The following variables with recursive definitions help to simplify the equations:

$$\begin{aligned}
\mathbf{z}^{(l)} &= \mathbf{W}^{(l)}\mathbf{a}^{(l)} + \mathbf{b}^{(l)}, \\
\mathbf{a}^{(l)} &= f^{(l)}(\mathbf{z}^{(l)}), \\
\mathbf{a}^{(0)} &= \mathbf{x}.
\end{aligned}$$

Provided these definitions, we can see that $g(x) = \mathbf{a}^{(3)}$. In implementation, it is useful to store these variables while we process a forward pass, as we will need the terms again in the backward pass.

In backpropagation during the training procedure, we do not simply only calculate the result of Equation A.13, but also calculate the value of the loss function, or in fact, as we shall see, of its derivative. Let this loss function denoted $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$. We then want to calculate each of the partial derivatives $\frac{\partial \mathcal{L}}{\partial w_{ij}}$, with w_{ij} being the weight of the connection between neuron i of one layer and neuron j of the following. Namely, the entire vector of partial derivatives make up the *gradient* that indicates the direction of greatest change of the loss function at the location of interest. Remember that the chain rule can briefly be explained in the following way: if a variable f depends on a variable a that on itself depends on a variable x, then $\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\mathrm{d}f}{\mathrm{d}a} \cdot \frac{\mathrm{d}a}{\mathrm{d}x}$ [89]. The backpropagation principle is best explained when we consider the partial derivative with respect of $\mathcal{L}$ with respect to the input $\mathbf{x}$ instead of the individual weights, which we finally actually need. Namely, applying the chain rule to this calculation, we obtain an expression that has many intermediate terms that are full derivatives, from which we can then calculate the partial derivatives with respect to each of the weights:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\mathbf{a}^{(3)}} \cdot \frac{\mathrm{d}\mathbf{a}^{(3)}}{\mathrm{d}\mathbf{z}^{(3)}} \cdot \frac{\mathrm{d}\mathbf{z}^{(3)}}{\mathrm{d}\mathbf{a}^{(2)}} \cdot \frac{\mathrm{d}\mathbf{a}^{(2)}}{\mathrm{d}\mathbf{a}^{(2)}} \cdot \frac{\mathrm{d}\mathbf{z}^{(2)}}{\mathrm{d}\mathbf{a}^{(1)}} \cdot \frac{\mathrm{d}\mathbf{a}^{(1)}}{\mathrm{d}\mathbf{z}^{(1)}} \cdot \frac{\partial \mathbf{z}^{(1)}}{\partial \mathbf{x}}. \tag{A.14}$$

The $\frac{\mathrm{d}\mathbf{a}_i}{\mathrm{d}\mathbf{z}_i}$ terms are the derivatives of the activation function $f^{(i)\prime}(\mathbf{z}^{(i)})$, while the $\frac{\mathrm{d}\mathbf{z}^{(i)}}{\mathrm{d}\mathbf{a}^{(i-1)}}$ terms are exactly equal to $W^{(i)}$. The first two terms correspond to the derivative of the loss function with respect to $\mathbf{z}^{(3)}$, $\mathcal{L}'(\mathbf{z}^{(3)}, \mathbf{y})$, which also involves the derivative of the activation function of the third layer, $f^{(3)\prime}(\mathbf{z}^{(3)})$. Often, the pair of activation function and loss function are chosen in such a way that this derivative corresponds to a trivial calculation. For example, in case of combining the Sigmoid function for $f^{(3)}(\mathbf{z}^{(3)})$ with the binary cross-entropy loss function for $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$, where $\hat{\mathbf{y}} = \mathbf{a}^{(3)}$, it can be shown that these two terms together correspond to $\mathbf{a}^{(3)} - \mathbf{y}$. We will assume those functions in the remainder of this section to simplify the terms further. Following, we explain how we can use these terms to calculate the gradients of the weights.

Although Equation A.14 corresponds to the partial derivative, we actually want to obtain the gradient. The gradient and the derivative are related in the sense that the gradient is simply the transpose of the vector that is the derivative. It can therefore simply be obtained by transposing the weight matrices and changing the order of the terms [96]. Here, however, we will continue to explain the procedure from the partial derivative perspective, which will eventually lead to the same values and is more intuitive, as we can read it from left to right. In the case of the gradient expression, we would have to multiply from right to left instead. With the assumptions and replacement of

the terms as indicated in our explanation below Equation A.14, the expression becomes as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = (\mathbf{a}^{(3)} - \mathbf{y})\mathbf{W}^{(3)} f^{(2)\prime}(z^{(2)})\mathbf{W}^{(2)} f^{(1)\prime}(z^{(1)})\mathbf{W}^{(1)}. \tag{A.15}$$

Before we can evaluate this expression, we will have to rearrange them to ensure that the dimensions of the matrices align. Note that we will not actually perform the last multiplication, as we are interested in the partial derivative with respect to the input itself, but use the intermediate terms because they also appear in the expressions of partial derivatives with respect to each of the individual weights. These common terms are in fact the derivatives with respect to each layer's vector of scaled and translated input values $\mathbf{z}^{(l)}$. We will equivalently denote these terms $\delta^{(l)}$. The algorithm thus evaluates this expression with a single term at a time, which corresponds to traversing the network backwards, starting at the deepest layer, storing each $\delta^{(l)}$ in between. From those values, we can calculate $\frac{\partial \mathcal{L}}{\partial w_{ij}}$, with node j being in layer l, as $\frac{\partial \mathcal{L}}{\partial W^{(l)}} = \delta^{(l+1)}(a^{(l)})^T$. Note that the bias values are also included in this, as we recall from Figure A.6 that the biases are in fact simply common weights assigned to the bias node of the previous layer, and are therefore multiplied with the constant value of 1 as indicated by that node. The formula of this term, denoted $\delta^{(l)}$, lends itself nicely to an intuitive recursive definition [96]:

$$\delta^{(l-1)} = f^{(l-1)\prime}(\mathbf{z}^{(l-1)}) \circ (\mathbf{W}^{(l)})^T \delta^{(l)}. \tag{A.16}$$

Note that here we corrected the order to align the dimensions of the matrices. The vector δ_l contains the derivatives with respect to $z^{(i)}$, which were the values of $\mathbf{W^{(i)}}$ multiplied by $a^{(i-1)}$, as we recall from Equation A.13.

From Equation A.16, an important problem with neural network training also becomes clear: the *vanishing gradient* problem. Several loss functions have gradients that yield values between 0 and 1. If we propagate such values backwards and use them in the chain rule multiplication, we would persistently end up with smaller values each time we propagate the error further backwards, i.e. $\delta^{(l-1)} < \delta^{(l)}$, thus decreasing drastically. Once the value becomes infinitesimally small, the earlier layers in the network may not be able to learn at all any more from that point onwards. Notice that small gradient values do not only depend on the weights and the derivative of the activation function, but also on the input values, which is where batch normalisation layers could help. There is also the opposite effect, where the gradient can *increase* as it is propagated backwards. This can happen with loss functions of which the gradients yield values greater than 1, in which case the gradient could in fact

increase exponentially as we propagate the error backwards.

Additionally to reusing the already calculated value of $\delta^{(l)}$ in our calculation of $\delta^{(l-1)}$, as argued above, backpropagating the error instead of propagating the error forwards through a multiplication starting at an earlier layer, translates itself to a multiplication of the vector $\delta^{(l)}$ by the weight matrix by the derivative of the loss function, compared to the matrix by matrix multiplication we would obtain by multiplying forwards instead [96]. Note that the specific loss functions and activation functions do not matter in backpropagation, as long as they are differentiable and the gradient can preferably be calculated in an efficient manner. After calculating the gradient with respect to each weight individually, the weights are then updated accordingly to steer the network towards a lower loss value.

An important note to make in this section is that the revolutionary idea of CNNs, namely that of *shared weights* for convolutional layers, is applied in this step. Namely, even though the gradients are calculated for the weights of the synapses correspondonding to each pixel individually, these are then summed and averaged across the depth slice to update shared weights only once [48].

A.9 Gradient Descent

After thus obtaining the partial derivatives with respect to each of the weights individually, a specific optimisation algorithm determines how exactly the calculated gradients are used to update the weights. Commonly, the gradient descent algorithm is used for this purpose. Although many variations on this algorithm now exist and have been shown to perform better than the generic version under certain circumstances, the basic concept of gradient descent applies to them all. Intuitively, we can visualise the solution space of the loss function or cost function of our network as a manifold, which we may imagine as a hilly landscape in a simplified 3D space, as visualised in Figure A.7. Gradient descent is an algorithm that then searches to minimise the function by repeatedly taking a step into the direction of the steepest descent as indicated by the negative of the gradient that we obtained from the backpropagation. In Figure A.7 this process is visualised for a 3D solution space, which is visualised as the aforementioned hilly landscape. We also show the step-wise solution the gradient descent algorithm would find, parting from the shown (random) starting point, as we explain below. Note that this simplified example only has a single minimum, while in reality, problem spaces can have many local minima.

Gradient descent only has information about the gradient, and thus the direction of the greatest change, of the loss function at the current point. We can imagine this value to represent the steepness of the hill in our 3D solution landscape. All gradient descent knows, is that it wants to perform a parameter update that will take us a step lower on the hilly landscape, towards a minimum. As we do not have any further information about the shape of the often complicated manifold (an often used analogy is that our hilly solution landscape is misty), all we can do is take a step into the direction of the negative gradient towards a lower point. However, what should the optimal size of this step be? This step size that the gradient descent algorithm should use at any point in time thus becomes an important parameter, often referred to as the *learning rate*. It is commonly multiplied by the values of the negative gradient with respect to each weight to update those weights and finally take a step into the direction of a minimum. In this way, the learning rate essentially allows us to control the overall time gradient descent requires to converge to a minimum, also called the convergence time.

However, there are some issues with the learning rate that are important to consider. The risk of setting this parameter too high is that we may *overshoot* the minimum. As the gradient will be relatively large at those locations of our solution space where the slope is relatively steep, a high learning late on such a location could cause us to constantly overshoot the minimum, i.e. to jump out of the region of valley in our solution space, and make our gradient descent fail to converge. In this figure we plotted our loss function in a 2D-space for illustrative purposes. If, on the other hand, we use a too low learning rate, not only will our algorithm take unnecessarily long to converge as we take tiny steps on surfaces with a shallow slope, but at the same time it could cause us to end up with a very weak local minimum that gradient descent would ignore (overshoot) with a more appropriate learning rate. In the basic version of gradient descent, we usually have to make this trade-off at the start of our learning procedure, although heuristics can be employed to change the learning rate throughout the learning procedure. In fact, many of the later variants on gradient descent implement an method to intelligently adjust the learning rate automatically. We will discuss these and other methods in the remainder of this section, although we will first discuss different variants of gradient descent based on how much data they consider in a single update.

With respect to the amount of update considered in a single update, we can distinguish between three variants of gradient descent: batch gradient descent, mini-batch gradient descent and stochastic gradient descent [73]. Batch gradient descent computes the gradient on the entire training set at once before

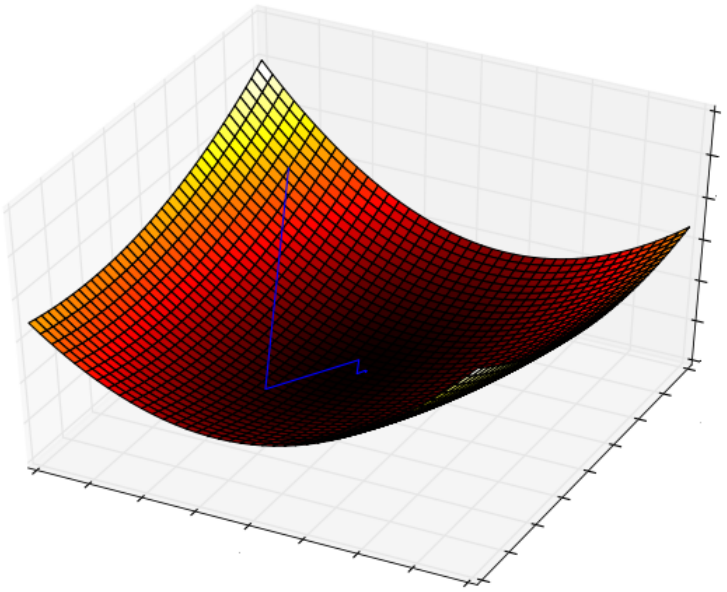


Figure A.7: A generic example of the gradient descent algorithm, with its step-wise solution visualised as a blue line, for a 3D problem space.

it performs a single update to the weights. With this method, it is necessary to load the whole dataset into memory, which for most datasets nowadays is infeasible, while it also converges very slowly due to performing redundant computations in large datasets where gradients are recomputed for similar examples as weights are not changed in between. On the other extreme, we have stochastic gradient descent (SGD), which performs a parameter update for every single training training example. Although this avoids the redundant computations in batch gradient descent for large data sets, relatively big differences between subsequent training samples causes significant fluctuations in the cost function throughout the training procedure. On the one hand it is sometimes argued that this can help SGD jump out of a relatively high local minimum and converge to the global or a lower one, while on the other hand it could even cause SGD to overshoot important minima, or even the global minimum, during the learning procedure. Choromanska et al. show, however, that for sufficiently large datasets, local minima are likely comparably good and at least as good as global minima, as adjusting to the global minima tends to lead to overfitting [12]. Therefore, the variant that is the middle ground between the two and is often considered to have best of both worlds, is mini-

batch gradient descent. This variant instead calculates the gradient for a small batch of predefined size n of input samples, which can be adjusted to the data set and the available memory as needed. If the batch size is perfectly adjusted to the data set to contain a combination of samples that is representative for the data set, this method would be the most efficient memory-wise, time-wise and solution-wise. This is therefore also most popular variant in the state of the art. It is also referred to as "SGD with a batch size of n".

Apart from the update procedure, gradient descent algorithms further vary based on the heuristics they employ, which are designed to help overcome certain inconveniences of naïve gradient descent. Most of these aim at varying the learning rate throughout the learning procedure. In our work using CNNs to detect intestinal content, we made use of Momentum, Adam and Nadam and will therefore briefly explain each of them.

A.9.1 Momentum

Momentum is a method designed to help SGD navigate better towards the minimum when one of the parameters contributes significantly more to the gradient than the others, which we can imagine as being around a ravine on our simplified manifold, where the surface decrease much more steeply into one direction than into others. While naïve gradient descent would only slowly take steps towards the minimum oscillating around the slope of the ravine, this method accelerates and improves that procedure by always adding a fraction of the previous update vector to the current one [73]. The fraction to be used is a parameter of the momentum method, often simply referred to as *the momentum* in SGD.

A.9.2 Adam

The next method, Adam (Adaptive Motion Estimation), is a method that computes adaptive learning rates for each of the input features in accordance with their prevalence. This idea was not new, as it was first introduced by Adagrad and improved in Adadelta and its equivalent method RMSprop. The prevalence of the features is determined through the magnitude of the gradients, whereas a greater magnitude is associated with a lower prevalence, for which the learning rate is adjusted to higher values. Adam uses the improved method of using a decaying average of past squared gradients. The concept

that Adam newly introduces is that of using an exponentially decaying average of past gradients that is similar to the idea of momentum. In our earlier example of going down a ravine with momentum behaving as a heavy ball, Adam would behave as a heavy ball with friction and thus prefer flat minima in the error surface [73].

A.9.3 Nadam

The last method that we used in our work is Nadam proposed by Dozat [20]. This method improves on the Adam method with the same idea as the Nesterov-accelerated gradient (NAG) applied to the original momentum method. Namely, NAG attempted to look ahead to save a future step, by calculating the gradient over the value of the parameter that has already been adjusted for the momentum term. After that, the momentum is calculated as before and the parameter value is adjusted accordingly. Dozat does not only propose to apply this concept to Adam to create their Nadam method, but at the same time proposes to modify the concept by, instead of calculating the gradient over an already adjusted value, applying the look-ahead momentum vector directly to the parameter adjustment. Both the gradient and the momentum are then calculated as normally, while instead of updating the parameter value with the current momentum value, we update its value with a look-ahead momentum vector, which adjusts the calculated momentum with the momentum proportion and again sums the gradient already adjusted by the learning rate.

Index

activation function, 137, 158
active contours, 106
anchor box, 80, 84, 94
annotation, 12
average precision, 89

backbone, 76
backpropagation, 135, 163, 165
batch normalization, 152
bounding box regression, 86

CNN architectures
    AlexNet, 43
    Inception, 44
    ResNet, 46
    VGG, 47, 66
Cohen's kappa, 32
colour spaces, 34
    CIELAB, 35, 105
    RGB, 10, 34
convolutional layer, 146
    convolution, 140
convolutional neural network, 42
cross-validation, 26, 161
    nested, 27

frame rate, 10, 11
fully convolutional, 150

gradient descent, 167

hand-crafted features, 33
    local binary patterns, 36

intersection over union, 85
intestinal content, 21
intraclass correlation coefficient, 31

largest inscribing circle, 104
loss function, 88, 153
lumen, 72

manometry, 70, 101
model evaluation
    accuracy, 28
    MCC, 28
    sensitivity, 28
    specificity, 28
model selection, 27
morphological snakes, 105
motion estimation, 101

overfitting, 25, 52, 139

padding, 147
patches, 23
pooling, 82, 150

R-CNN, 75
region of interest
    alignment, 84
    pooling, 82
    quantisation, 83
    warping, 84
region proposals, 79
    region proposal networks, 80
    RPN, 81
    selective search, 79

segmentation, 106
sliding window, 22
softmax, 153
stride, 147
surrogate model, 27
SVM, 39
    kernel trick, 40

transfer learning, 43

tunnel, 72

undersampling, 52

YOLO, 92, 95