UiT THE ARCTIC UNIVERSITY OF NORWAY

Faculty of Science and Technology

Department of Physics and Technology

Investigating the Effect of Class-Imbalance on Convolutional Neural Networks for Angiodysplasia Detection

—

Oskar Anstein Kaspersen

FYS-3921 Master's thesis in electrical engineering 30 SP - October 2019

Abstract

Angiodysplasia (AD) is a serious problem among patients older than 50-60 years that may cause both acute gastrointestinal bleeding and chronic iron deficiency anemia. Accurate detection and localization of AD lesion for early-stage diagnostics is important, but most of the small intestine is out of reach for traditional endoscopy because it can only reach a fraction of the small intestine. This changed due to the development of wireless capsule endoscopy, were a capsule is swallowed and records images while it passes through the whole digestive system for several hours. However, in order to analyze the large amount of image data, automatic approaches need to be developed to assist the medical expert during his task.

In this thesis, we look at a deep learning approach to AD detection and focus specifically on the problem of class imbalance, which arises from the fact that lesions only occupy a small part of the images, by analyzing how weighting of the loss function can help address this issue. Balancing the weights of the foreground and background class in the cost-function was found to be crucial to achieve good segmentation results.

Acknowledgments

Above all, I would like to express the deepest appreciation to my supervisors Robert Jenssen and in particular to Michael C. Kampffmeyer for the effort and guidance in this thesis, and for the assistance in all aspects of this thesis.

Thanks to my fellow students at the master office.

Thanks to work colleagues for patience and help.

Thank to my family and friends for help, support and consideration.

Oskar Kaspersen
Tromsø, October 2019

Contents

List of Tables

List of Figures

Section With Acronyms

Angiodysplasias (AD)

adaptive moment estimation (adam)

Batch Normalization (BN)

charged-coupled device (CCD)

capsule endoscopy (CE)

complementary metal oxide semiconductor (CMOS)

Convolutional Neural Network (CNN)

chronic renal feilure (CRF)

Deep Learning (DL)

Deep neural networks (DNNs)

Fully Connected (FC)

Fully Convolutional Network (FCN)

false negative (fn)

false positive (fp)

Gastrointestinal (GI)

Gradient Descent (GD)

graphics processor unit (GPU)

ground truth (lGT)

hidden Markov model (HMM)

Hue, Saturation, and Value (HSV)

irritable bowel syndrome (IBS)

lighting memory-mapped database (LMDB)

learning rate (lr)

Median Frequency Balancing (MFB)

Obscure GI bleeding (OGIB)

Rectified Linear Unit (ReLU)

Suspected Blood Indicator (SBI)

true positive (tp)

true negative (tn)

Wireless capsule endoscopy (WCE)

Chapter 1

Introduction

1.1 Angiodysplasia disorder in the bowel

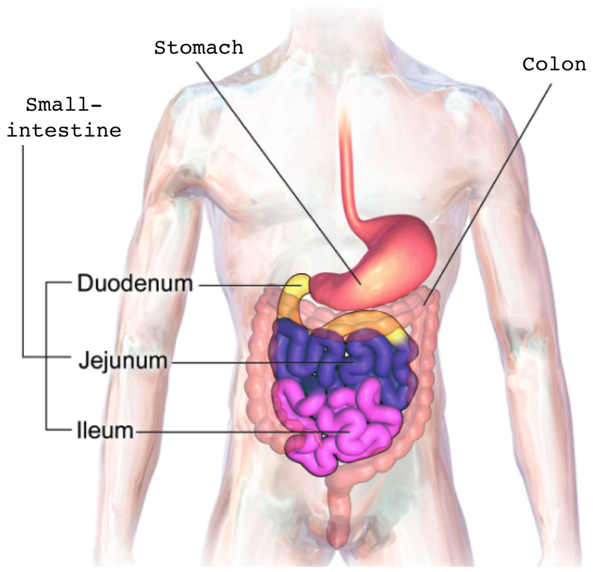


Figure 1.1: The gastrointestinal tract[1]. Total length for humans about 8.5m.

Disorder in the bowel can arise at any time in life, and one symptom which gives cause for concern is to have blood in the stool. Patients experiencing this will normally contact their doctor for a referral of an examination to

[1]Source: https://sml.snl.no

a medical expert in order to exclude other serious diseases such as cancer. Angiodysplasia (AD) occurs when degenerative lesions of previously healthy blood vessels are formed. They have a cherry red color, form vascular lesions of 2 to 10 mm in size on the bowel wall and can cause both acute gastrointestinal (GI) bleed and/or chronic iron deficiency anemia if left untreated. The wound usually has a round shape and is shown in Figure 1.2. Not all AD bleeds and the probability of bleeding depends on the lesion shape and size. When it increases in size, the risk of bleeding is higher (Vieira, Gonçalves, Gonçalves, & Lima, 2016).

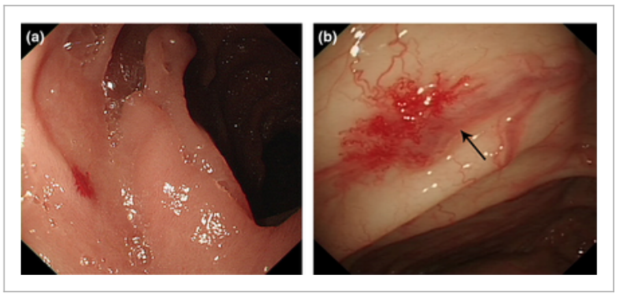


Figure 1.2: (a) Illustration of AD in small boval. (b) Illustration of AD in the ascending colon, an arrow indicates the feeding vessel. Both images are diagnosed by endoscopy (Sami et al., 2014).

1.2 Need for automation

Doctors or the gastroenterologist experts have various examinations available for the diagnostics of GI bleeding and anemia. Early-stage detection and accurate localization for AD lesions is essential in early stage diagnostics (Shvets, Iglovikov, Rakhlin, & Kalinin, 2018). One common approach for the diagnosis of the GI tract is to perform a screening or examination using flexible video endoscopy. This means sending a flexible tube into the intestine for visualization and taking tissue samples from the mucous membranes, see Figure 1.3. The figure illustrates a typical endoscopic video rack. To perform examinations, the flexible endoscopy equipment, colonoscope, or gastroscope (depending on the type of examination) connects to the video rack, and the camera feed is fed to the video monitor. The endoscope is equipped with lights, and normally a HD-camera on top of the flexible distal end. The doctor (endoscopist) controls the video endoscope front movements of the visual field by means of the steering wheel with the hand control knobs (shown by circles in the figure), and the flexible distal end, and takes images and tissue samples among other things.

The total length of the whole GI tract is about 8.5 meters. The colon surrounds the small intestine like a frame, see Figure 1.1. Until a few years ago, most of the small intestine was out of reach of endoscopic techniques due to its length, curvature and several complex loops and standard endoscopes (such as gastro and colon) could only reach a fraction of the small intestine (Manno et al., 2012). Nowadays there are other endoscopic approaches that can be used. One is the wireless capsule endoscopy (WCE).

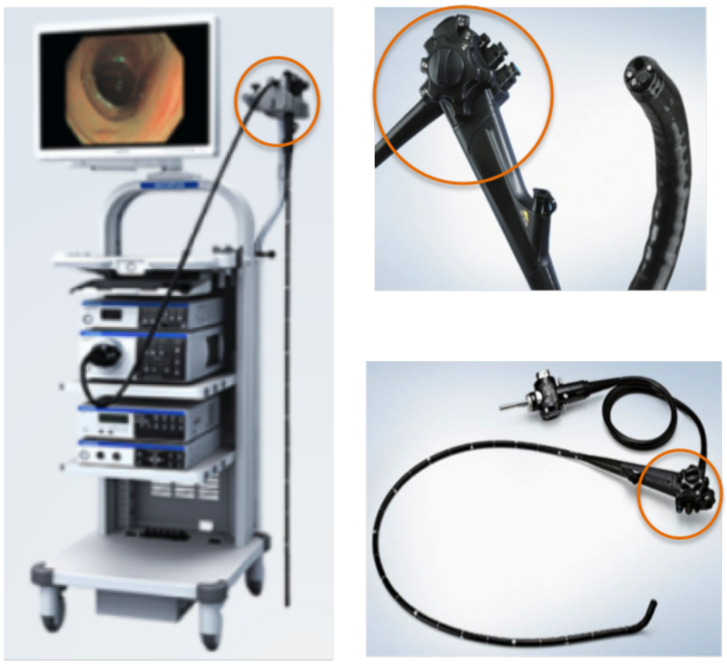


Figure 1.3: The figure illustrates on the left a typical endoscopic videorack and illustrates on the right a flexible endoscope and the steering wheel. The use of image copies is in agreed with Olympus, https://www.olympus-europa.com.

WCE is a small pill device that is swallowable and is the first-line investigation for the small intestine in the context of obscure GI bleeding (OGIB) (Vieira et al., 2016). The WCE captures high resolution images during the passage through the whole digestive system. An image of the pill-capsule and a typical WCE bowel image is shown in Figure 1.4. However, experience shows that investigating WCE videos becomes an expensive and time-consuming procedure. Even for an experienced viewer it can take more than an hour to analyse each WCE video (Maieron et al., 2004). On the other hand, recent studies have shown that the gastroenterologist experts discover only 69% of AD's when reading the WCE videos (Vieira et al., 2016).

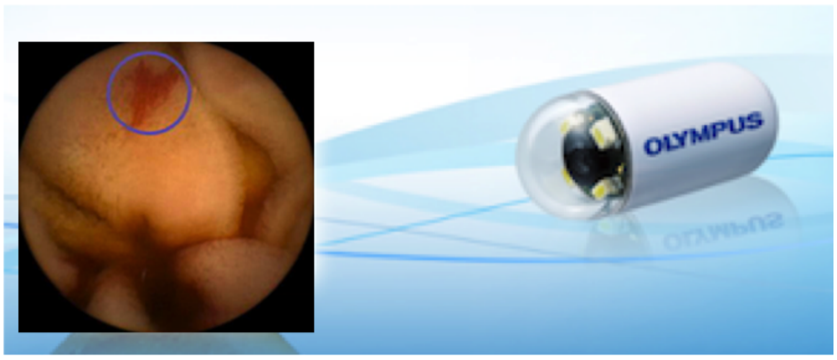


Figure 1.4: Example of a pill-capsule (right), and a WCE image captured from capsule endoscopy (an angiodysplasia lesion in the circle). The use of image copies is in agreed with Olympus, https://www.olympus-europa.com.

1.2.1 Blood Indicator Software

Lesions can be missed due to a variety of reasons, for example a lack of experience of the gastroenterologist, poor visibility, and size and location of AD, in particular if they are covered behind mucosal folds. A computer-aided blood indicator software made by WCE providers (like Given Imaging) can assist in the decision making process, but the present sensitivity and specificity values are only 41% and 67%, respectively (Vieira et al., 2016). As a consequence, there is a real need to improve the analyses of video capsule endoscopy software. One promising direction to achieve this is machine learning, in particular deep learning.

1.3 Machine learning

Machine learning is a field that resides in the intersection between mathematics, statistics, and computer science. It is considered to be a subcategory of the over-ordered concept of artificial intelligence, as machine learning allows intelligently behaviors to be learned from observing the surrounding environment through data. The basics of machine learning have been reviewed by

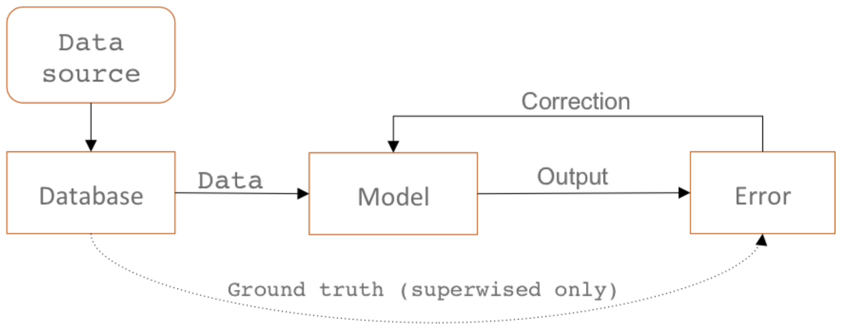


Figure 1.5: An illustration of a universal machine learning system.

several textbook authors (Shalev-Shwartz & Ben-David, 2014), (Theodoridis & Koutroumbas, n.d.), and (Nielsen, 2015), and although there are some variations in formulation and terminology, these authors all succeed in conveying the essence of the machine learning field. In particular, machine learning algorithms or models are designed that can be trained to perform specific tasks, without them being explicitly programmed with an exact task in mind. Instead the model learns by being presented with training examples, also often referred to as training data, and is therefore able to adapt to a variety of tasks.

In all areas of society, there is a continual state of digitalization, and the generation and collection of huge amounts of data are a natural consequence and outcome of this process. The design and use of machine learning is therefore absolutely necessary as the size of the datasets has long ago exceeded the possibilities of manual analysis and explicit programming. Inspired by this, researchers have been designing increasingly complex models to handle the ever-growing volume of data.

Figure 1.5 shows a simplified overview of a machine learning system. The data is collected and transferred to the model, which then generates a set of outputs. The outputs are processed by a cost function or a loss function with the purpose to quantify the error produced by the system. For supervised learning (see next section), the training data consists of sequences of input-output pairs. Each input is labeled with the desired output value, in this way the system is trained to perform a minimization of the particular loss function by mapping a given input to the desired output. The output examples are referred to as ground truth or labels.

One discriminates between supervised and unsupervised learning (Theodoridis & Koutroumbas, n.d.):

- In supervised learning, the training data consist of input-output pairs, and the system is trained to determine the correct output for each input.
- In unsupervised learning, the training data only consists of input observations. The training data is therefore referred to be "un-labeled". Due to the lack of output examples, a loss function needs to be designed that is based only on the underlying structure of the data.

1.4 Deep learning

Deep learning is a subfield of machine learning that has recently found application on a large variety of tasks, both supervised (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012), (Krizhevsky, Sutskever, & Hinton, 2012), and unsupervised (Xie, Girshick, & Farhadi, 2016), (Kampffmeyer et al., 2019).

One of the first papers that paved the way towards deep learning as it is known today, was published by McCulloch and Pitts (Cheng & Titterington, 1994), and proposed a simple computational model that is often referred to as an artificial neuron. Research on artificial neurons continued to be carried out throughout the 1960s (Gurney, 1997), but the absence of modern computers put a damper on their development until the interest in artificial intelligence increased in the 1980s, when neural networks, a hierarchy of neurons, were applied for recognition tasks. (Rumelhart, 1986) and (LeCun et al., 1989). Enabled by the availability of computational resources, large-scale datasets and new training methods, artificial neural networks, also referred to as deep learning, started to find widespread adaption in 2012 (LeCun, Bengio, & Hinton, 2015).

1.4.1 Deep Learning for semantic segmentation

The focus of this thesis is on the task of supervised semantic segmentation, an important task for scene understanding. The objective of semantic segmentation is to assign each pixel in the image to a given class, in order to obtain coherent regions and finds application for task such as self-driving cars (Fernandez-Moral, Martins, Wolf, & Rives, 2018), (Bojarski et al., 2016), and land cover mapping (Kampffmeyer, Salberg, & Jenssen, 2016). Also in the medical imaging domain there has been a huge interest in the semantic pixel-wise labeling task e.g. (Akbari et al., 2018) proposed polyp segmentation in colonoscopy images using fully convolutional network, (Bernal, Sánchez, & Vilarino, 2012) proposed an automatic polyp detection with a polyp appearance model, and (Ronneberger, Fischer, & Brox, 2015) proposed Convolutional networks for biomedical image segmentation using the U-Net, all examples of deep learning based segmentation methods that obtained state-of-the-art performance. However, one problem of common deep learning models for the task of segmentation, especially in the medical domain, where objects of interests are commonly small compared to the background class, is class imbalance. In particular, class imbalance can lead the model to prioritize large classes at the cost of small ones (Shvets et al., 2018), (Akbari et al., 2018), resulting in bad performance for the class of interest.

1.5 Scope

It is important to detect angiodysplasia wounds in the gastrointestinal tract based on images from video capsule endoscopy, a time-consuming task if performed manually. There exists therefore a need for an automatic based approach for accurate detection of AD in order to reduce time and costs.

In previous work deep learning models have been proposed to perform image segmentation for AD detection and have achieved state-of-the-art performance (Shvets et al., 2018), (Vieira et al., 2016), however, a more thorough analysis of the effect of class-imbalance on the models for AD segmentation is lacking. Here we seek to asses the problem of imbalanced classes and aim to analyze the effect of introducing a weighting scheme to alleviate the problem of class imbalance.

1.6 Structure Of The Thesis

After this introduction, we will proceed to Part 1, which covers relevant background in Chapters 2, 3 and 4. Part II covers the remaining part of the thesis and includes the experimental section and the conclusion.

Chapter 2 provides a medical introduction to wireless capsule endoscopy and angiodysplasia in particular.

Chapter 3 provides a more through background of the machine learning techniques used in this thesis. We start the discussion from linear classifiers and the perceptron and discuss how muliple perceptrons can be used to obtain non-linear classifiers. A detailed explanation of neural networks is provided, followed by an explanation of convolutional neural networks and a discussion of how they can be used to address segmentation problems.

Chapter 4 provides a brief discussion of how deep learning has found application in the medical domain. Special attention is put on work related to angiodysplasia segmentation.

Chapter 5 presents the results obtained in this thesis and an analyses of the weighting scheme as well as additional hyper-parameters.

Chapter 6 presents future work and concludes this thesis.

Part I

Background

Chapter 2

Endoscopy for Angiodysplasia detection

This chapter will discuss the use of endoscopic equipment for examination of the gastrotestinal (GI) tract and provide a brief overview of the relevant medical background. Video endoscopes are the main tool for detecting vascular lesions such as AD in the GI tract in the general population. In recent years, there has been a significant improvement in video endoscopy equipment and image resolution. The medical examinations of the GI tract are increasingly performed using endoscopic equipment. Until a few years ago, most of the small intestine was out of reach of endoscopic techniques due to its length, approximately 7 m plus the colon approximately 1.5 m.

2.1 Angiodysplasia

Angiodysplasia (AD) is the most common source of vascular lesion in the GI tract in the general population. This condition is mostly discovered in patients that are older than 50 years, but can occur earlier especially for patients with chronic renal failure (CRF). AD occurs when degenerative lesions of previously healthy blood vessels are formed. The abnormalities appear usually as small ($< 10mm$) bleedings visualized within the mucosa and sub mucosa layers of the gut (Sami et al., 2014). The condition may be asymptomatic for the patient or it may cause GI bleeding and or anemia, see Figure 2.1.

In a review, AD was identified among 11.9% of 642 patients that were diagnosed with irritable bowel syndrome (IBS) and among 12.1% of patients without IBS (Akhtar, Shaheen, & Zha, 2006). A systematic review was performed by (Liao, Gao, Xu, & Li, 2010) and included a large number of articles related to small bowel signs and symptoms published between 2000 and 2008. A total of 227 studies with a total of 22840 procedures were included. The review confirms OGIB (Obscure GI bleeding, overt and occult) as the most common indication (66.0%) and AD as the most common cause (50.0%) of bleeding in those patients (Liao et al., 2010). Another more recent study, found that small bowel AD lesions were the most common cause (35%) of severe life-threatening overt OGIB (Lecleire et al., 2012). AD can affect any part of the GI tract, and according to (Sami et al., 2014) the colon is the most frequent site of AD lesion. In western patients, lesions occuring in the caecum and ascending colon (54-81.9%), while lesions diagnosed in Japanese patients are more likely to be in the descending colon (41.7%). These findings suggest that local factors may be important to find out how the decease occurs and develops (pathogenesis) (Sami et al., 2014).

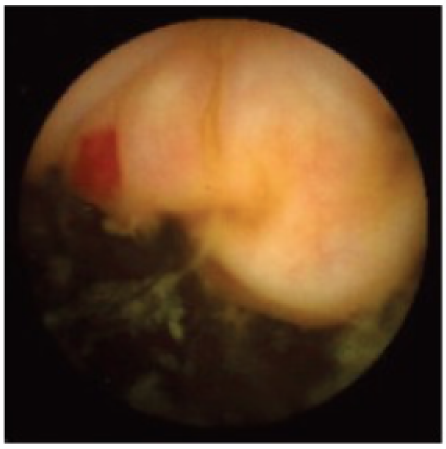


Figure 2.1: Example of angiodysplasia sore image taken by pill-capsule (Karagiannis et al., 2006).

2.2 Flexible video endoscopy

Video endoscopes are, due to high resolution cameras, the main tool for detecting even the smallest vascular anomalies. In order to diagnose upper GI and colon, respectively, standard flexible endoscopy (gastroscope and colonoscope) are used. One examination means sending a flexible endoscope into the intestine and take tissue samples from the mucous membranes. Endoscopic access in the small intestine is more troublesome and out of the reach for a standard endoscope, but there are other endoscopic approaches that can be used.

For examination of the small intestine, alternative approaches intended for deeper insertion are required (Sami et al., 2014) such as:

- Single Balloon Enteroscope (SBE)
- Doubleballoon enteroscopy (DBE)
- Spiral enteroscopy (SE)

Despite the rapid technological advances, it is still more difficult to take advantage of the enteroscopy than the upper gastrointestinal endoscopy or the colonoscopy, due to the length of the small intestine Figure 2.2 shows the principle of reaching into the small intestine (bowel) with the enteroscope by inflating and deflating the balloon. The balloon overtube provides deep insertion technique for the enteroscope and a control unit (which is not shown) inflates the balloon stop the enteroscope from going backwards. For example, the SBE endoscopists is able to reach 50-80% of the small intestine, but there are still parts of the small intestine that are difficult to reach using deep insertion techniques[1]. The use of flexible endoscopy and enteroscopy provides high-resolution image quality as well as excellent maneuverability and can be used to obtain tissue samples.

2.3 Wireless capsule endoscopy

Standard endoscopes and colonoscopes can only reach a fraction into the small intestine, however, the use of Wireless Capsule Endoscopy (WCE) among other techniques has recently enabled endoscopic examination of the entire small intestine. This is in particular due to significant advances in the video capsule equipment as well as image resolution. WCE is a small pill

[1]https://www.olympus-europa.com/medical/en/Products-and-Solutions/Medical-Specialities/Gastroenterology/

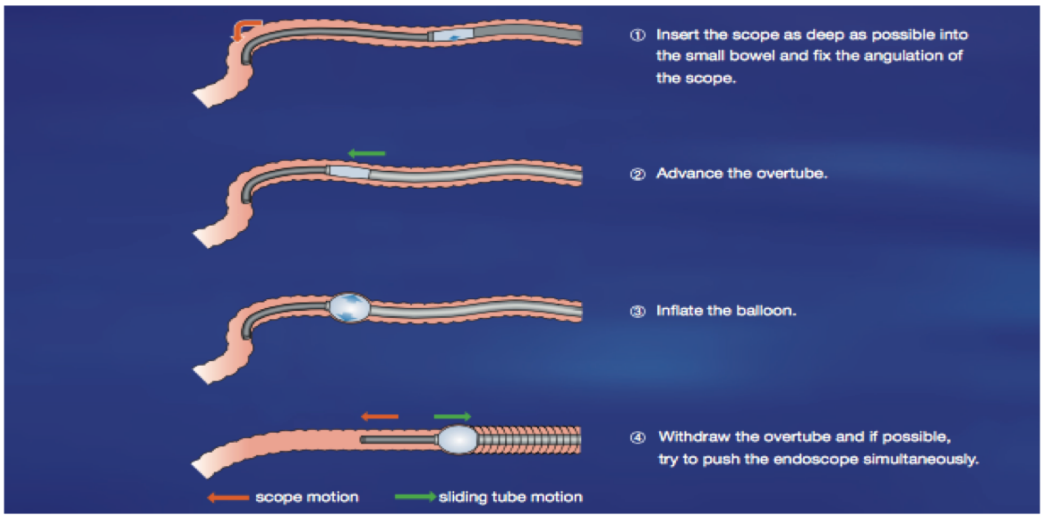

① Insert the scope as deep as possible into the small bowel and fix the angulation of the scope.
② Advance the overtube.
③ Inflate the balloon.
④ Withdraw the overtube and if possible, try to push the endoscope simultaneously.
scope motion
sliding tube motion

Figure 2.2: Single balloon enteroscopy. Simplified principles of insertion. https://www.olympus-europa.com/medical/en/Products-and-Solutions/Medical-Specialities/Gastroenterology/

device that is swallowable and has become the first-line investigation for the small intestine in the context of OGIB (Vieira et al., 2016). Last generation WCE can capture approximately 8 hours of video producing more than 60 000 frames during one passage through the whole digestive system.

Since their introduction in 2000, several capsules have appeared on the marked and there has been a continous development of capsule technology. A set of typical pill-capsules can be found in Figure 2.3.

These technological advances have made it possible to examine the whole GI tract and close the diagnostic gap of conventional gastroscopy and colonoscopy techniques in a non-invasive manner. Studies have shown that the WCE technique has improved the diagnostic outcomes among a variety of GI conditions and has given us diagnostic information for a variety of clinically relevant lesions in the digestive tract (Nadler & Eliakim, 2014). The WCE modality is today considered as the gold standard method (Shvets et al., 2018) because it provides a complete visual overview of the entire digestive system. It is also likely to be the preferred screening technique in the future, as it is considered to be safe and without any discomfort to the patient. (Iakovidis & Koulaouzidis, 2015). WCE is the only medical device that can visualize and capture images inside the entirely GI tract (Vieira et al., 2016).

The main drawbacks of the WCE is that no direct intervention can be per-

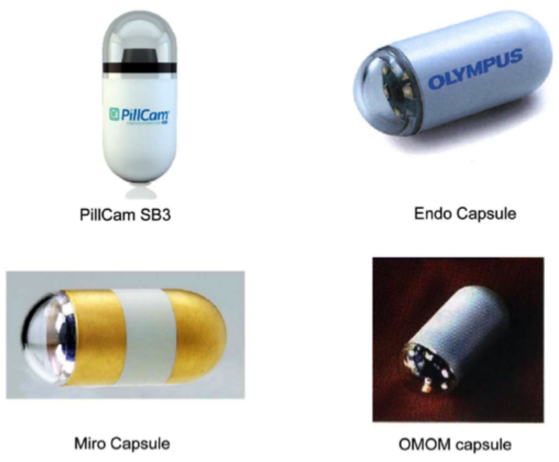


Figure 2.3: Capsule endoscope pills from several manufacters (Nadler & Eliakim, 2014).

formed and the poor visibility in the small bowel due to dark intestinal liquid contents. However, compared to other more invasive modalities, WCE has shown to achieve equivalent performance for discovering lesions (Vieira et al., 2016).

Bovel obstruction is one of the most feared complications but the risk of bowel obstruction is minimal , the retention rate is as low as 1.4% according to a systematic review (Liao et al., 2010).It may be higher for patients with underlying pathological complications who must be reviewed by a specialist before a capsule endoscopy (CE) procedure is decided. (Van de Bruaene, De Looze, & Hindryckx, 2015).

2.3.1 Pill-capsule

The capsule weighs between 3 and 4 gram and measures about 11 mm (diameter) x 26 mm (length). It includes an optics (lens) with a field of view from 140 to 170 degree. The light source consists of 4 to 6 white LEDs to illuminate the digestive tract system. It has a high-resolution miniature camera and sharpness with minimum detection size of 0.07 mm. Two types of camera chips are used for focusing the image, either a charge-coupled device (CCD) or a complementary metal oxide semiconductor (CMOS). The capsule camera takes 2-6 images per second and can capture more than 60 000 images during 8 to 13 hours as the capsule moves along the GI tract and the images are transmitted via radio signals to an external image receiver (Hosoe et al., 2016) (Van de Bruaene et al., 2015). The battery allows for more than 12 hours of operation. See Figure 2.4 for a schematic representation of a pill-capsule.

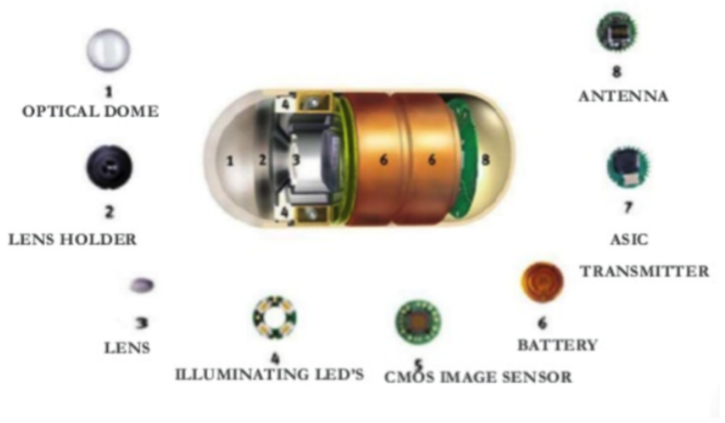


Figure 2.4: The pill-capsule, a schematic representation. Ref: www.slideshare.net

2.3.2 Examination

A capsule acquisition kit is worn by the patient during the aqusition. The kit consists of a single-use wireless video capsule, a belt-style antennas or a sensor array attached to the patient, and a smart data recorder receiver attached to the belt. All transmitted images are sent wireless and stored on the receiver. After the examination, the equipment is handed over to the doctor who can download all images to a computer for further processing to find abnormalities in the GI tract (Z. Li et al., 2014).

In order to obtain the best image results, it is important that all equipment is verified and tested by the medical staff, and that the patient is supervised and prepared before the examination begins. A brief outline of the CE procedure is as follows:

- All necessary equipment is verified and tested by the medical staff before the examination.

- The patient fasts the last 12 hours before the CE examination in order to obtain a pure (empty) bowel, which is of great importance to get a good result. Only the consumption of water is allowed during this time.
- The patient is equipped with an eight-sensor batch and a sensor-belt which is attached to the body.
- The capsule is swallowed, followed by a two-hour period of no fluid intake. Clear liquids may be consumed approximately 2 hours after capsule ingestion.
- After 4 hours, the fasting can end.
- The location of the pill-capsule in the bowel is tracked, and a 3D model is generated in order to map a particular image to a specific location in the body to help locate the lesions.
- After about 8 hours the recording is complete and will be reviewed by the medical expert.

2.3.3 Analysis

After the WCE examination the images are returned to the doctor, who downloads them to a computer for further analysis. Without any tool at hand to assist, it would take a gastroenterologist expert several hours to go through the image material. However, providers of the WCE usually have appropriate tools to help physicians by providing for example a suspected blood indicator (SBI) for a given image. The SBI feature is intended to mark frames of the video containing blood or red area. However, the detection rate of AD is small when using SBI (Vieira et al., 2016), and there is therefore a need for more accurate detection approaches based on machine learning.

Chapter 3

From linear classifiers towards deep learning models for image segmentation

In this chapter, we will provide the background on machine learning and deep learning that is required to build automatic decision support tools for AD detection. We will start by providing an explanation of the underlying idea of supervised learning, that our model builds on, by looking at a simple linear classifier, namely the perceptron algorithm. We then build on the perceptron algorithm and discuss how multiple perceptrons can be combined to model non-linear relationships resulting in a description of multilayer feedforward neural networks and their training procedure. Further, building on this, the convolutional neural networks, a deep learning model that commonly finds application in computer vision tasks, is introduced and an overview of how such a networks can be used for segmenting images is provided. A method for dealing with class imbalances in segmentation tasks is also illustrated.

3.1 Fundamentals of supervised machine learning

In supervised machine learning, the task is to, based on provided input-output pairs learn a parameterized function or a model that maps a given datapoint to its given output. The common approach to this is that a loss function (or cost function) is defined that measures the badness of the model, meaning how bad our model is at performing the mapping for our data. The objective of supervised machine learning is to design algorithms that based on this loss function can optimize or learn the parameters of the model. Given a trained/learned model, the idea is that new, previously unseen, datapoints can be mapped using the learned model. Note, it is not guaranteed that a model that works perfectly on the training data will also work perfectly on the unseen data and how well a model generalizes to new data depends on the model as well as the training process. Ways of improving generalization will be discussed later in this chapter.

3.2 Linear classifier

A perceptron model performs a weighted sum of input patterns $[x_1, x_2, ...x_n]$ in an n-dimensional room and sends out a binary "1" if the sum exceeds a threshold, in the opposite case it outputs a "0". These binary responses may be interpreted as regions in a multidimensional space that are separated by a straight line or a plane, and for higher dimensional input space separated by hyperplane.

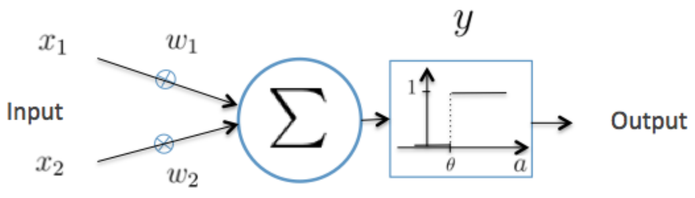


Figure 3.1: Two-input perceptron equipped with activation output threshold.

Figure 3.1 illustrates a 2-D example of a perceptron model. Here, the perceptron is a computational unit where inputs x_1 and x_2 are fed and multiplied by the respective weights ω_1 and ω_2, represented by circles (also known as synaptic weight). The underlying concept and principles of the perceptron will be covered more closely later in this chapter after the following initiating example. The perceptron model can be illustrated with an example where the threshold $\theta = 1.5$, $\omega_1 = 1$ and $\omega_2 = 1$. The resulting products are added to give an activation a, and then compared with the threshold value θ, to provide the output. The binary response (y=0 and y=1) are provided by a threshold based step function $[y = 0]$ if $a < \theta$ and $[y = 1]$ if $a > \theta$.

x_1	x_2	Activation a	Output y	Class
0	0	0	0	B
0	1	1	0	B
1	0	1	0	B
1	1	2	1	A

Table 3.1: The input and responses for a two-input perceptron module.

Note, that we here make use of a threshold (step function) that decides when a neuron is activated. This function is commonly referred to as an activation function, see subsequent Subsection, and will be later replaced by smooth approximations of the step function in order to keep the function that the model is learning differentiable. For now

$$f(x) = \begin{cases} 1 & x > \theta \\ 0 & x < \theta \end{cases} \tag{3.1}$$

The decision plane in Figure 3.2 illustrates that the datapoints are correctly classified to their corresponding classes and separated geometrically by a decision line.

Rectified linear activation function

The sigmoid activation functions were for a long time the preferred activation function used for neural networks. However, in order to learn more complex nonlinear structures in the data, modern deep learning approaches tend to make use of a large number of layers. In this setting, the sigmoid function has a drawback and leads to a problem that is commonly referred to as the

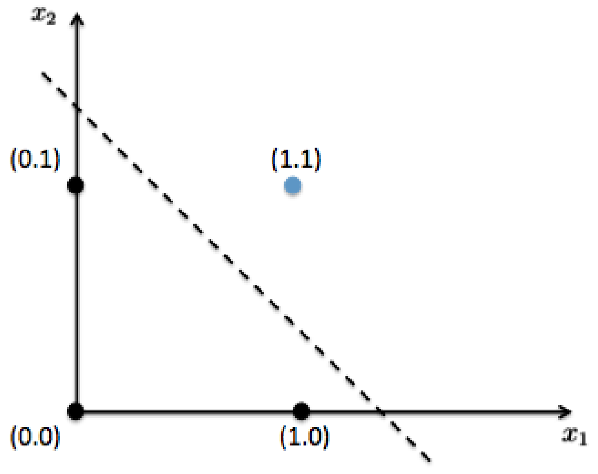


Figure 3.2: Two input patterns with a line between the two classes.

vanishing gradient problem. For details on the vanishing gradient problem, see the Subsection 3.4.4. One of the key factors to the success for deep neural networks (DNNs) is the adoption of a new activation function that reduces the vanishing gradient problem, the Rectified Linear Unit (ReLU) (Nair & Hinton, 2010). By considering a single value x, the ReLU is defined as

$$f_{Relu}(x) = \begin{cases} x & x > \theta \\ 0 & x \leq \theta \end{cases} \tag{3.2}$$

We see that positive values are returned regardless of their size, whereas negative values become zero. The derivative $f'_{Relu}(x)$ is

$$f'_{Relu}(x) = \begin{cases} 1 & x > \theta \\ 0 & x \leq \theta \end{cases} \tag{3.3}$$

The derivative $f'_{Relu}(x)$ is always equal to one for positive inputs, this by-passes the vanishing gradient problem. Note, that when $x = 0$, the derivative is not defined because of the discontinuity of $f'_{Relu}(x)$, which at first sight appears problematic. However, based on the theory of subgradients it is possible to compute the subderivative at the point $x = 0$ as $f'_{Relu}(x)$ is a convex function and thereby circumvent the problem. Valid subgradients for the point lie in range [0,1] and for simplicity 0 is commonly chosen.

3.3 Perceptron

In this section, we will provide a brief but more rigorous description of the perceptron algorithm starting from its underlying biological motivation until the Rosenblatt perceptron algorithm.

The perceptron is essentially an algorithm that aims to find a linear decision plane that separates the data. It is an online classifier in the sense that it updates the learned model based on single data points and can be updated continuously until the solution is found. It was initially proposed by Rosenblatt in the late 50's and was mainly intended to model how neurons behave in a certain way in the brain.

The perceptron is based on the idea of punishment and reward. If the classifier is correct it is rewarded by performing no update of the decision line, while the classifier is penalized for incorrect classifications in form of a correction of the learned model. A finite number of training and label datapoints enter the algorithm. The algorithm stops when all training data have been classified correctly.

The Widrow-Hoff or least mean squares (MSE) algorithm was proposed in the 60's to train the linear classifier. The weights in the Widrow-Hoff algorithm are updated as follows

$$\boldsymbol{\omega}(k) = \boldsymbol{\omega}_{(k-1)} + \rho_k \boldsymbol{x}_k [y_k - \boldsymbol{x}_k^T \boldsymbol{\omega}_{(k-1)}] \tag{3.4}$$

where:

y_k and $\boldsymbol{x}_k$ are the label output and the input training data.
$\boldsymbol{\omega}(k)$ are the weights at iteration k.
ρ_k is the learning rate.

The choice of ρ_k is important for the convergence speed of the algorithm and is often chosen to be a pre-defined hyperparameter, commonly found via cross-validation. Widrow-Hoff classifier updates the decision line for each new datapoint in order to find the weight configuration that minimizes the sum of squares of the errors. Errors are represented by the difference inside of the bracket $[y_k - \boldsymbol{x}_k^T \boldsymbol{\omega}_{(k-1)}]$ and by assuming that the threshold now is 0, thus if the argument $(\boldsymbol{x}_k^T \boldsymbol{\omega}_{(k-1)})>0$, then $\boldsymbol{x}_k$ belongs to class 1, else $\boldsymbol{x}_k$ belongs to class 2. The Widrow-Hoff algorithm is often referred to as an iterative process because each new data point $\boldsymbol{x}_k$ leads to an updated weight vector $\boldsymbol{\omega}$.

In general, when having the loss function and when updating the model by finding the derivative with respect to the weights. By looking at a single datapoint at a time, a rough estimation for the MSE-classifier, we have that the derivative of the cost function in step (k-1) is the expectation $E[\boldsymbol{x}(y - \boldsymbol{x}\boldsymbol{\omega}^T_{(k-1)})]$ (Theodoridis & Koutroumbas, n.d.).

The Rosenblatt perceptron algorithm is a little different from Widrow-Hoff because Rosenblatt included an activation function in his training algorithm that will change the way we look at the inner-product within the bracket. Instead of assigning a class just by checking if the argument $\boldsymbol{\omega}^T\boldsymbol{x}$ is positive or negative, here a step function assign $\boldsymbol{x}$ to class 1 if $f_{step}(\boldsymbol{\omega}^T\boldsymbol{x}) = 1$ and assign to class 2 if $f_{step}(\boldsymbol{\omega}^T\boldsymbol{x}) = -1$ (Theodoridis & Koutroumbas, n.d.). The step function assigns the feature vector in one of the classes according to the two possible output value, and thereby brings outputs and labels within the same domain to [1 and -1]. The weight update rule for the Rosenblatt perceptron algorithm becomes

$$\boldsymbol{\omega}(k) = \boldsymbol{\omega}_{(k-1)} + \rho_k \boldsymbol{x}_k[y_k - f_{step}(\boldsymbol{x}_k^T \boldsymbol{\omega}_{(k-1)})] \tag{3.5}$$

The Rosenblatt classifier does nothing when the algorithm receives a new training datapoint where $\boldsymbol{x}_k$ and label $y_k = 1$ because $\boldsymbol{x}_k$ is on the positive side of the decision line. The second segment of the algorithm becomes zero which states that the new solution = old solution. In a similar situation, Widrow-Hoff would have moved the decision line. This means that if all future training datapoints are correctly classified the current decision line will stay fixed and the model will not be updated continuously.

The XOR function

Unfortunately, there are some issues with the simple perceptron algorithm. A critical analysis of the perceptron algorithm in 1969 by Minsky and Papert, pointed out that the perceptron can not do nonlinear classification such as for example the simple logical operation XOR (exclusive or), as one of many drawbacks with the perceptron (Sathiyabama, 2007). The XOR problem is illustrated in Figure 3.3, where the two classes are illustrated with black and blue data points. Note, the data points can not be separated by a straight line in this case.

x_1	x_2	XOR	Class
0	0	0	B
0	1	1	A
1	0	1	A
1	1	0	B

Table 3.2: Truth table for the XOR function.

A logical XOR function will always produce 1 output if either of the two binary input values x_1 and x_2 is 1, and will 1 produce a 0 output when both of its inputs are 0 or 1. Table 3.2 gives the analogous truth table for the XOR operations.

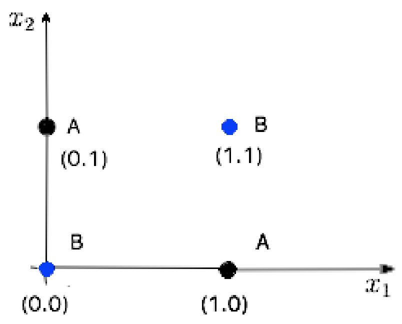


Figure 3.3: Placement of classes A and B for the XOR problem.

In the following section, we will show how a combination of perceptrons can be used to address this problem and model non-linear structure in data.

3.3.1 Two layers perceptron

One approach to address the XOR problem and to extend the model to more general nonlinear separable cases is to first project the datapoints into a new feature space where the datapoints are linear separable and can be separated by a linear decision line. In order to do this, multiple perceptrons can be combined to obtain the final result. Intuitively, what we would like to achieve can be observed in Figure 3.4, where the data points of the two classes A and B are separated by two decision lines, g_1 and g_2, instead of one, where datapoints that fall in-between the two decision lines are classified to class A and the remaining points to class B. As hinted at earlier, we can obtain this by making use of two perceptrons, one modelling the boolean AND function and one modelling the OR function. Table 3.3 shows the truth table and the appropriate class position. Note, both of these functions can be represented by a single linear decision line (a single perceptron).

x_1	x_2	AND	Class	OR	Class
0	0	0	B	0	B
0	1	0	B	1	A
1	0	0	B	1	A
1	1	1	A	1	A

Table 3.3: Truth-table for the AND and OR functions.

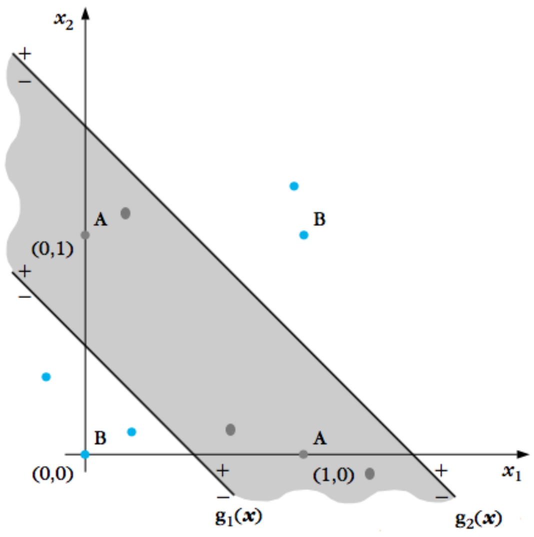


Figure 3.4: Decision lines to solve for the XOR problem (Theodoridis & Koutroumbas, n.d.).

Following the proposed mapping with help of our two perceptrons, we now receive a binary output indicating on which side a datapoint lies for each of the two decision plane. This means that for each datapoint we receive two binary input values, here denoted as y_1 and y_2. From Figure 3.4, we see that the two data points in the upper right are on the positive side of both the decision lines g_1 and g_2 and will therefore map to the point [1,1] in our new feature space. Similarly, data points that are on the negative side of both g_1 and g_2 are mapped to the point [0,0]. Performing this mapping results in the truth table mapping illustrated in Table 3.4, and the new feature representation in Figure 3.5. Note, after the transformations the two classes are now linearly separable in the obtained feature space and can be easily separated using an additional perceptron, now acting in the obtained feature space.

x_1	x_2	y_1	y_2	2nd phase
0	0	0(-)	0(-)	B
0	1	1(+)	0(-)	A
1	0	1(+)	0(-)	A
1	1	1(+)	1(+)	B

Table 3.4: Truth table for the computation of the outputs $[y_1, y_2]$ in the second phase.

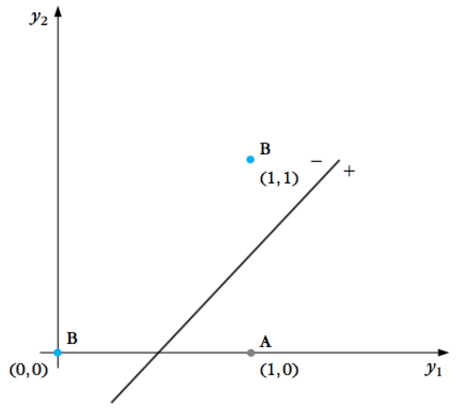


Figure 3.5: Decision line for y-room (Theodoridis & Koutroumbas, n.d.).

Figure 3.6 shows a model consisting of the steps in both phases. The two neurons (nodes), represented by the lines g_1 and g_2, are from the first layer which is often referred to as the so-called "hidden layer". The single neuron of the second (output) layer computes the (so-called output). Such a multilayer network consisting of multiple perceptrons is known as a two layers perceptron or a two-layer feedforward neural network.

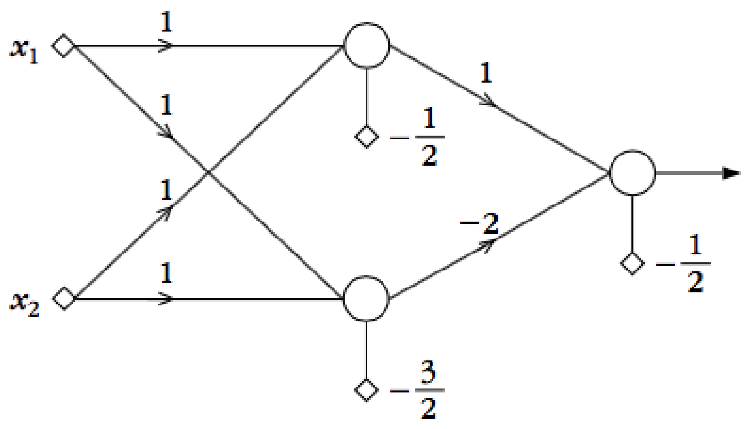


Figure 3.6: Solving the XOR problem with applying the two-layers perceptron (Theodoridis & Koutroumbas, n.d.).

3.3.2 Gradient descent

To obtain a low value of the cost (error) function is a goal we want to achieve, and the derivative of the error function with respect of the $\boldsymbol{\omega}_{(k-1)}$ stipulates a position in an iterative process. To iteratively direct along an error function towards the minima is a process where the gradients of the error function are estimated, and then used to update the parameters. The update equation for the Widrow-Hoff, which uses the Euclidean norm for the distance measurement provides the equation as defined

$$\boldsymbol{\omega}_{(k+1)} = \boldsymbol{\omega}_k - \rho_k \boldsymbol{x}_k(\boldsymbol{\omega}_k \boldsymbol{x}_k - y_k), \tag{3.6}$$

where:

ρ_k is the learning rate.

The ρ_k plays a vital part on the convergence. If ρ_k is too small, then the correction $\Delta\omega$ is low, and the converge will go very slowly to the optimal point. If on the other hand, it is too large, the algorithm may oscillate around the optimal value and not converge. One can improve the decision line position from step (k) to step (k+1) by adjusting the step length Δk in the direction of the gradient to the cost function $J(\boldsymbol{\omega}(k-1))$. Hopefully, once the parameters are correctly selected, the algorithm converges to a stationary point of $\boldsymbol{\omega}(k)$, which can be either a local minimum or a global minimum or a saddle point, as shown in Figure 3.7. In other words, it converges to a spot on the curve where the slope is flat, and the gradient becomes zero. Which of the stationary points the algorithm will converge to depends on the conditions relative to the fixed points along the graph. Also, the convergence speed depends on the structure of the cost function (Theodoridis & Koutroumbas, n.d.).

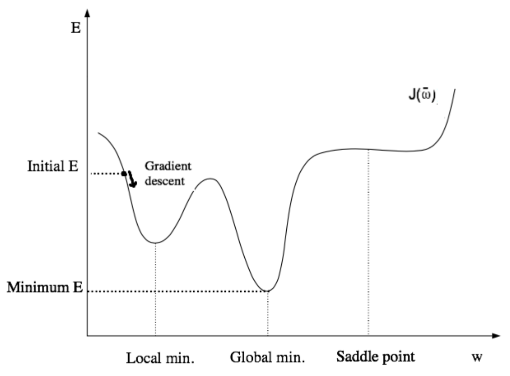


Figure 3.7: Cost function with local versus global minimum.

A test can confirm if the cost function is at a minimum. The underlying graph of the cost function is unknown, and there is a risk of being trapped in a local minimum point that we do not get out of, see Figure 3.7. We can try again with new start conditions. It might be possible sometimes to have an extra push to overcome intervening bumps such as between local and global minimum in the figure. If the system is not performing well, it may be due to a local minimum (Theodoridis & Koutroumbas, n.d.).

3.4 Feed-forward neural networks

The previous section has shown that a combination of perceptrons are expressive enough to model non-linear functions, such as the XOR problem. However, in order to make use of this fact, a learning algorithm is needed that can learn the weights in the individual perceptrons/neurons of our model from data. This will be the focus of the following section.

Feed-forward neural networks, or multilayer perceptrons, aim to learn the parameters in the network by minimizing the error/cost/loss (E), which can be represented by the difference between the desired target (the given label) and the prediction (output) of the network. The output of the network is a function of the weights and the objective is to find the weight configuration that minimizes the error. In order to do this, neural networks commonly make use of gradient-descent based algorithms, such as the one illustrated in Subsection 3.3.2. The motivation is still the same as for the XOR problem and the model aims to find a mapping to some feature space (through multiple transformations) where the data is ideally linearly separable.

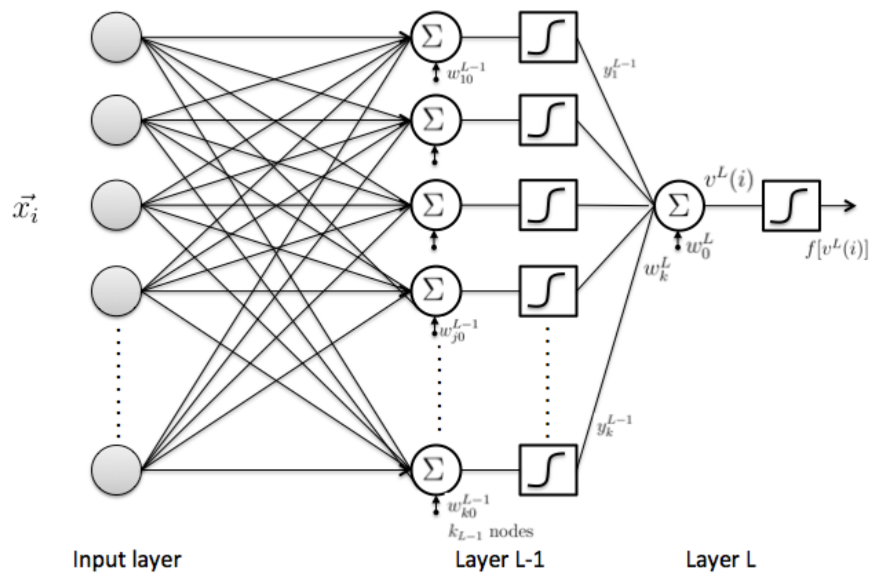


Figure 3.8: Feedforward neural networks with one hidden layer and sigmoid function in the output layer L.

The example network in Figure 3.8 is similar to the feedforward neural network published in Nature by (Rumelhart, 1986). The network has a set of inputs and one so-called hidden layer. The hidden layer, so-called because we do not have direct access to neurons outputs, is connected with its neurons to the inputs and they must develop their values on the input vectors. All nodes in the hidden layer are connected to a single node in the output layer. The network output layer corresponds to the two classes of prediction. In general, the input layer only passes data to the first hidden layer. Between the input and output layer, we can have one or several hidden layers. The hidden layers are responsible for transforming the weighted inputs, and apply the activation function for doing the mapping into representations, where the output layer can optimally perform the required task. For traditional feedforward neural networks this activation function is commonly the sigmoid function. More detail on the sigmoid function as well as an alternative activation function that is more commonly used will be discussed in Subsection 3.4.1.

The output layer signals the response of the network to any input and may have the label vector $\boldsymbol{y}(i)$ applied to it during the supervised training part. In general, this networks can be trained to separate two or more non-linear classes dependent on the number of neurons in the output layer. We assume a network with L layers, where the $L'th$ layer is the output layer consisting of k_L neurons, where each neuron is followed by an activation function. Together, the outputs of the k_L neurons will contribute to the sum of error $\epsilon(i)$ for each data point. With k_L neurons in the output layer resulting in

$$\epsilon(i) = \frac{1}{2}\sum_{m=1}^{k_L} \boldsymbol{e}_m^2(i) = \frac{1}{2}\sum_{m=1}^{k_L}[f((\boldsymbol{v}_m^L(i)) - \boldsymbol{y}_m(i)]^2 \tag{3.7}$$

For a binary problem, where the objective is to classify the input into two classes, the output layer only needs to contain a single node. The function $f(\cdot)$ inside the bracket in Equation (3.7), is the nonlinear sigmoid function that takes the output to one of the classes depending on positive or negative function argument. The difference between the output of the neural network $f(v_m^L(i)$ and the actual label $y(i)$ for a single $x(i)$, given by the $\epsilon(i)$, is the error from the network. We want that error to be small. Squaring will ensure errors being positive such that negative and positive errors do not counteract each other.

Network errors are summed in Equation (3.7) one by one, and the class is assigned.

Because there are only one neuron in the output layer, Equation (3.7) is simplified

$$\epsilon(i) = \frac{1}{2}\boldsymbol{e}^2(i) = \frac{1}{2}[f(\boldsymbol{v}^L(i)) - \boldsymbol{y}(i)]^2 \tag{3.8}$$

The overall error

$$J = \sum_{i=1}^{N} \epsilon(i) \tag{3.9}$$

The node in the output layer will do the classifying job. We update the network for each node by finding the derivative of the cost function based on the unknown weight $\boldsymbol{w}^{L-1}$ for each node in the hidden layer $(L-1)$. The number of weights is the same as the number connections since each connection has a node and $\boldsymbol{y}_k^{L-1}$ is the output for the neurons in the (L-1) layer, with k_{L-1} nodes. Based on the gradient for the cost function, we use an iterative Widrow-Hoff update. Thus, the output from the L-neuron, but before the activation function $f(\cdot)$ is

$$\boldsymbol{v}_j^L(i) = \sum_{m=1}^{k_{L-1}} \boldsymbol{w}_{jk}^L \boldsymbol{y}_k^{L-1}(i) + \boldsymbol{w}_{j0}^L = \sum_{m=0}^{k_{L-1}} \boldsymbol{w}_k^L \boldsymbol{y}_k^{L-1}(i) \tag{3.10}$$

The number of nodes in the hidden lager is k_{L-1} and $\boldsymbol{w}_k^L$ are the weights between the hidden layer and the output layer, and $\boldsymbol{y}_k^{L-1}(i)$ are the outputs from the hidden layer. In the continuation of this derivation, $\boldsymbol{w}_0$ is included in the sum.

It is wise and necessary not to derivative the actual cost function (J) because it is the error $\epsilon(i)$ in Equation (3.9) that comes for each of the training datapoints $x(i)$ we wish to work with. We use the method sum rule in differentiation, such that when we find one derivative, we sum the derivatives covering the cost function. By now we use the chain rule of differentiation, which is

$$\frac{\partial}{\partial \boldsymbol{w}_j^L}\epsilon(i) = \frac{\partial}{\partial \boldsymbol{w}_j^L}\boldsymbol{v}_j^L(i)\frac{\partial \epsilon(i)}{\partial \boldsymbol{v}_j^L(i)} \tag{3.11}$$

The gradients in the network are found by computing the first part of the expression in Equation (3.11). To find the outputs from the previous (L-1) we derivative the $\epsilon(i)$ with respect on the weight vector $\boldsymbol{w}_j^L$, one output $y_k^{L-1}(i)$ for each node and they are collected in a vector $\boldsymbol{y}^{L-1}$. This new vector is expanded with the bias $\frac{\partial}{\partial}\boldsymbol{w}_0^L = 1$ and will contribute to the iterative Widrow-Hoff update. The vector $\boldsymbol{y}^{L-1}$ develops to input for the node in the output layer L.

The last derivative part in the expression in Equation (3.11) is the inner product from layer L, the output node, and before the sigmoid function and is the derivative of Equation (3.8) describing an error from the output node, and we denote this error $\delta^2(i)$, hence

$$\delta^L(i) = \boldsymbol{e}^2(i) f'(\boldsymbol{v}^L(i)) \tag{3.12}$$

Where f' is the derivative of the differentiable sigmoid function in the L layer). Now we have solved the error problem, along with unknowns for the node in the L layer, and can go further forward to layer L-1. By applying the chain rule once more, see Equation (3.11), we will understand that it is a pattern that makes it work in the same way ahead, we obtain

$$\frac{\partial}{\partial \boldsymbol{v}_j^{L-1}(i)}\epsilon(i) = \sum_{m=1}^{k_{L-1}} \frac{\partial}{\partial \boldsymbol{v}_m^L(i)}\epsilon(i)\frac{\partial \boldsymbol{v}_m^{L-1}(i)}{\partial \boldsymbol{v}_j^{L-1}(i)} \tag{3.13}$$

The errors in which the node do in output layer L affects the derivatives forward in the network and are needed to find error $\delta_j^{L-1}(i)$ in layer L-1. If there was a layer L-2, thus these errors propagate further in the network for all nodes in layer L-2. The network can be tested by following the Widrow-Hoff GD procedure, see the Equation (3.4) (Theodoridis & Koutroumbas, n.d.). Start by pre-initialize weights and vectors with small random values first, before the network is presented to training data, and update for each epoch. In online training, each datapoint will append an update of the network.

That is the basis of neural networks. The $f'(\boldsymbol{v}^L(i)$ is the derivative of the activation function. To summarize, the backpropagation algorithm can be performed by doing the following steps

1. Initialization: All weights are initialized using a weight initialization scheme (a procedure for restricting the network by initializing the weights closer to an ideal configuration).

2. Forward computations: Compare all inner product $v_j^r(i)$ and $y_i^r(i)$

3. Backward computation: Compute $\delta_j^L(i)$ and $\delta^{r<L}(i)$

4. Update all weights

5. Repeat 2-4 until convergence.

A typical used loss function L for neural networks is the sum of a squared error loss function

$$C = \sum_{i=1}^{N} \varepsilon(i) = \sum_{i=1}^{N} \sum_{m=1}^{k_L} e_m^2(i) = \frac{1}{2} \sum_{i=1}^{N} \sum_{m=1}^{k_L} (y_m(i) - \hat{y_m}(i))^2 \tag{3.14}$$

3.4.1 Activation function

One convenient choice to approximate the step function is the sigmoid function in Equation (3.15). This replaces the step function by a continuous squashing function, so that the output y builds smoothly on the activation parameter.

$$f(x) = \sigma = \frac{1}{1 + e^{-ax}} \tag{3.15}$$

The sigmoid function tends to better approximate the unit-step function for large positive values of a. It approaches 1 as the activation goes to infinity. Similarly, it approaches 0 for large negative values of a. Figure 3.9 shows the sigmoid curve properties with different values of a.

3.4.2 The Softmax function

The softmax activation function is applied to produce the final outputs in both neural networks and DNNs that perform classification. The softmax function is an extension of the sigmoid function to multiple classes, and

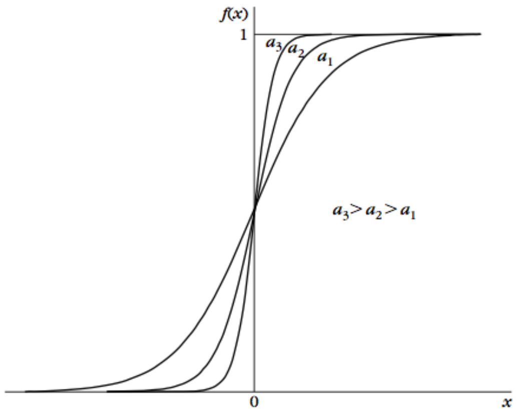


Figure 3.9: Example of the sigmoid curves for various values of a, along with the step function in Equation (3.1) (Theodoridis & Koutroumbas, n.d.).

takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one that sums to one. This provides a probability distribution over all the classes in the dataset. In this way, the network with L layers will assign datapoint into one of k classes. The softmax function is defined as

$$\hat{y_k} = \frac{exp(v_k^L)}{\sum_{k'} exp(v_{k'}^L)} \, , \; k = 1, 2, ..., k' \tag{3.16}$$

where:

L , is the number of layers
k' ,is the number of classes
v_k^L ,is the predicted class of the true class L for sample k
$v_{k'}^L$,is the predicted class score over k'

3.4.3 Methods making gradient descent faster

Gradient descent that corresponds to the Equation 3.6 incorporates all training feature or sample (x_i, y_i) in its weight update. However, an alternative is an abbreviate stochastic gradient descent method that updates the model based on a few randomly chosen training inputs. Updates are computed for a small set of samples (...) which commonly is referred to as a batch. This is an inherent averaging process for a small amount batch size of samples x_1, x_2. Stochastic gradient descent helps speed up training as not the full training set has to be processed for each individual update.

The following updated weight $\boldsymbol{\omega}_j^r(new)$ is based on the the randomly chosen mini-batch

$$\boldsymbol{\omega}_j^r(new) = \boldsymbol{\omega}_j^r(old) - \mu \sum_{i=1}^{N} \delta_j^r(i) y^{r-1}(i) \tag{3.17}$$

where:

$\boldsymbol{\omega}$ are the weights
μ is the learning rate
$\delta_j^r(i)$ is the gradient w.r.t. $\boldsymbol{\omega}$
$y^{r-1}(i)$ is the output of the neural network

In addition to stochastic GD the weight update Equation 3.17 is often modified to also include the weight update of the previous iteration. This in generally speeds up the convergence and can help to avoid bad local minima (n shallow networks). Adding the momentum term fraction α of the previous weight update accelerates the algorithm in the relevant direction. The weight equation becomes

$$\triangle\, \boldsymbol{\omega}_j^r = \alpha \triangle \boldsymbol{\omega}_j^r(old) - \mu \sum_{i=1}^{N} \delta_j^r(i) y^{r-1}(i) \tag{3.18}$$

where:

α is referred to as the momentum parameter.

If both weights contribute to updating toward the same direction the (GD) step size is amplified. Momentum smooths out changes of gradients and provides stability.

3.4.4 Vanishing and exploding gradients

Traditionally, training of DNNs has been affected by the vanishing gradient problem. This problem is mainly related to the fact that gradients in DNNs trained by the backpropagation algorithm are often affected by the vanishing gradient phenomenon.

The condition is mainly related to very deep networks where the sigmoid activation function is applied. The problem refers to the fact that gradients in deep neural network trends towards zero as the errors are propagated back through the network. This restricts the learning process and limits the size that networks can have for end-to-end training.

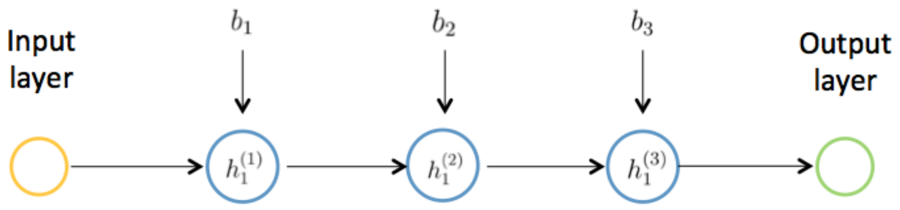


Figure 3.10: Simple neural network to illustrate why vanishing gradient problem occurs. the network has three hidden layers and one node in each layer, b_1, b_2, b_3 are the biases.

To understand why the vanishing gradient problem occurs, we consider the simple network example with three hidden layers in Figure 3.10. From Equation 3.12 we have for the output layer

$$\delta^r = e(i)\sigma'(v^r(i)) \tag{3.19}$$

where σ' is derivative of the sigmoid activation function. By proceeding to the second last layer we get

$$\delta^{r-1} = e^{r-1}(i)\sigma'(v^{r-1}(i)) \tag{3.20}$$
$$\delta^{r-1} = \delta^r \omega^L \sigma'(v^{r-1}(i)) \tag{3.21}$$
$$\delta^{r-1} = e(i)\omega^L \sigma'(v^r(i))\sigma'(v^{r-1}(i)). \tag{3.22}$$

To illustrate how the vanishing problem occurs we proceed further and we get

$$\delta^{r-2} = e(i)\omega^{L}\sigma'(v^{r}(i))\omega^{L-1}\sigma'(v^{r-1}(i))\sigma'(v^{r-2}(i)). \tag{3.23}$$

As gradients propagate further in the network, additional multiplicative terms of the form $\sigma'(v^{k}(i))\omega^{k}$ per layer are added. In practice, since the derivative of the sigmoid function reaches a maximum $\sigma'(0) = \frac{1}{4}$ this will cause the gradients to diminish due to the multiplicative terms as long as the weight are not large enough to counter the effect. However, as weights are commonly initialized with small values from a Gaussian centered around 0 weights tend to be small. This is done as large weights are likely to cause the opposite, the so-called exploding gradient problem. One of the important factors of DNNs recent success is the development of new activation functions with the ability to address the vanishing gradient problem, and this is still an active field of research.

3.5 Addressing the overfitting issue

Deep Neural Network architectures can have millions of parameters allowing a wide range of training conditions. In the case of classification (supervised learning), the parameters are learned based on a labeled training set (x_n, y_n). Even if the training details agree well with the available training data, it might not be valid when exposed for a test dataset, and this can lead to so-called overfitting. A sufficiently large network might memorize individual distinctions of the training data and achieving high accuracy without discovering the actual underlying distribution of the data that results in poor performance on the test set. The situation that the trained function perform well on the training dataset, and most importantly on new situations, is referred to as generalization requirement. Figure 3.11 displays a typical case of overfitting, where the training error continues to diminish while the test error starts increasing.

In order to reduce this undesirable effect and improve the generalization ability of the trained model, the overall loss function that is optimized tends to be a combination of a term that measures the goodness of the model on the training dataset and a term that regularizes the model to a more restricted set of solutions. The regularization term is often a function of the model weights and here denoted as r(w). One such regularization term is the L2 regularization commonly found in neural networks (also referred to as weight decay), which encourages the sum of squared weight parameters to be small, thereby encouraging weights to be small. A hyper-parameter (here denoted as α) is commonly added to control the balance between regularization and the model loss. This means that the regularized error function C is

$$C(\omega, b) = \sum_{i=1}^{N} \varepsilon(i) + \alpha \cdot r(\omega) \tag{3.24}$$

where:

ω , is the weight
b , is the bias
$\varepsilon(i)$, is the cost function
α , is the regularization parameter
$r(\omega)$, is the regularization term (weight decay)

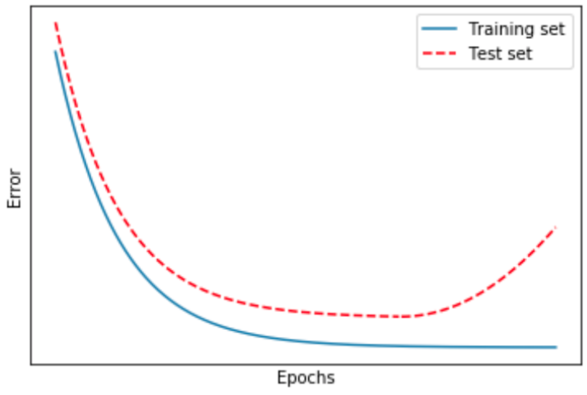


Figure 3.11: A representative example of overfitting. DNNs can have a large number of free parameters that can adapt to particular details in the training set which can lead to poor generalization performance. The two curves show the output error as a function of iteration steps. The training error keeps decreasing during training while the test error starts increasing.

This form of L2 regularization is also known as the Tikhonov regularization (Ng, 2004). This means that the cost function becomes

$$C_{L2} = \sum_{i=1}^{N} \varepsilon(i) + \frac{1}{2}\alpha \sum_{l=1}^{L} \sum_{k=1}^{k_l} \omega_{lk}^2 \tag{3.25}$$

where:

ω_{lk} , refers to the lk^{th} element of the weight vector $\boldsymbol{\omega}$
L , is the number of layers
k_l ,is the number of units in l^{th} layer

L2- regularization pushes the weights closer to the origin to restrain them from being too large. Choosing the α parameter is intuitively in favour of small values for the weights. Large weights will only be permitted if they considerably improve the first term of the cost function (Nielsen, 2015), but Hinton (Hinton, 2012) suggested the value 0.0001 for α as a sensible initial choice. Now computing the the partial derivatives $\partial C/\partial\omega$ and $\partial C/\partial b$ for all the weights and biases in the network. The partial derivatives of Equation (3.25) becomes

$$\frac{\partial C_{L2}}{\partial \omega_{jk}} = \frac{\partial \sum_{i=1}^{N} \varepsilon(i)}{\partial \omega_{lk}} + \alpha\omega_{lk} \tag{3.26}$$

$$\frac{\partial C_{L2}}{\partial b_{lk}} = \frac{\partial \sum_{i=1}^{N} \varepsilon(i)}{\partial b_{lk}} \tag{3.27}$$

where:

b_{lk} ,refers to the lk^{th} element of the bias vector $\boldsymbol{b}$.

From Equations (3.26) and (3.27) The $\partial C/\partial\omega$ and $\partial C/\partial b$ can be calculated using backpropagation. We see from Equations (3.27) that the partial derivatives with respect to the biases are unchanged, and hence the GD learning rule for the biases are not changing from the usual rule. The learning rule for the weights becomes

$$\omega_j^l(k) = \omega_j^l(k-1) - \mu \frac{\partial C_{L2}}{\partial \omega_j^l} = (1-\mu\alpha)\omega_j^l(k-1) - \mu \sum_{i=1}^{N} \varepsilon(i) \tag{3.28}$$

Equation (3.28) shows exactly the same as the usual gradient decent learning rule, except we first multiply the weight ω by a factor $1 - \mu\alpha$. The last term in (3.28) may influence the weights to increase, however, then it will cause a decrease in the unregularized cost function (Nielsen, 2015). Another popular technique is the L1 regularization to be derived in a similar way.

In the following section, we will discuss another commonly used regularization technique in more detail, namely Dropout.

3.5.1 Dropout

Achieving a good performance for the training set, especially in large networks with a large number of weights and biases is a trivial task. However, getting these models to generalize well is another problem. A powerful regularization method that aims to address this is Dropout.

For each iteration of the training procedure, dropout randomly turns off units of neurons (and their connections) to prevents neurons from co-adapting, see Figure 3.12. However, it is not only a technique that prevents overfitting but also encourages each hidden neuron to learn meaningful features without relying too much on other hidden neurons in the layers. More precisely, dropout can be interpreted as a form of ensemble learning, where a small sub-network is trained at each iteration and all small sub-networks are combined during the inference phase. Note, this is possible due to the fact that weights in the small sub-networks overlap (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).

During the test phase, dropout is not being used and all the neurons in the model are active. However, to ensure that the model works as intended, the weights of the model have to be rescaled. This means that if a given unit is present with probability p during training then the outgoing weights are scaled by a factor p. With dropout a significantly better generalization can be obtained (Srivastava et al., 2014).

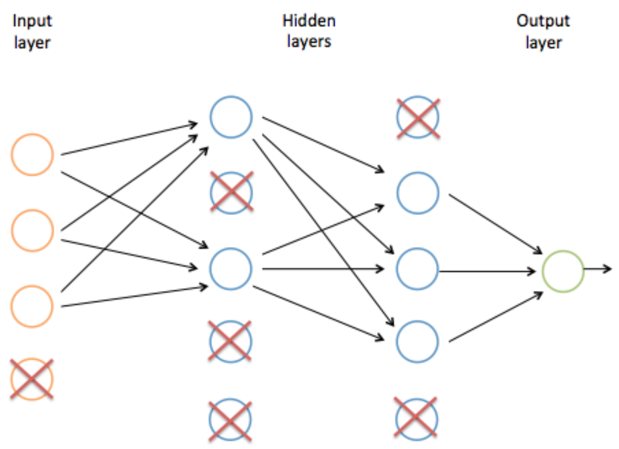


Figure 3.12: Shows the effect of dropout applied on a simple neural network. During training, dropout produces a thinner net where connections between subsequent layers are randomly removed.

A modified variant is the inverted dropout[1], where scaling is performed during training. This has the benefit that the test phase does not have to be changed and no scaling has to be performed during testing.

3.5.2 Batch normalization

Batch normalization (BN) is another technique commonly found in DNNs and even though it is commonly thought of as a normalization technique it can also be interpreted as a specific form of regularization.

Originally, BN was proposed to reduce the effect of internal covariate shift (Ioffe & Szegedy, 2015). This effect refers to the fact that small changes in parameters in early layers can get amplified and lead to large changes in activations in the layers above. This can cause further problems due to the fact that extreme activations can lead to saturation of neurons, which will impact the training speed. BN aims to minimize this effect by normalizing the activation of the hidden neurons so that the distribution of these activations stay almost constant. This is achieved by adding BN at the various stages in your architecture.

BN scales the activation for each mini-batch so that the values are centered

[1]http://cs231n.github.io/neural-networks-2/#reg

around 0 and have a unit variance. This has shown to dramatically accelerate training rates and ensures that the distribution of the inputs to the non-linearities do not get stuck in the saturated regime. Additionally, BN also reducing the dependence of gradients initial values which allows for much higher learning rates without the risk of divergence (Ioffe & Szegedy, 2015).

By looking at a mini-batch b of m examples with the values $x_1 ... x_m$ where $\mu_{\mathbf{b}}$ and $\sigma_{\mathbf{b}}$ represent the mean and standard deviation within a batch of samples for each neuron independently, the computation of BN takes the form

$$\hat{x_i} = \frac{x_i - \mu_{\mathbf{b}}}{\sigma_{\mathbf{b}}} \tag{3.29}$$

where:

$\hat{x_i}$,is the normalized value of the ith hidden unit.
$\mu_{\mathbf{b}}$,is the sample mean of $x_1 ... x_m$.
$\sigma^2_{\mathbf{b}}$,is the sample variance of $x_1 ... x_m$.

Just normalizing each input of a layer constrains what the layer can represent. In (Ioffe & Szegedy, 2015), they solve this by learning for each of the neurons activations one scale parameters γ_i and one shift β_i, one for the mean and the other for the variance which in practice allow the model to "unlearn" the normalization if it is desirable to achieve better training performance. These parameters are learned and updated for each hidden layer just like the weights and biases during training and shift the normalized value of the inputs. The transformation to y_i is

$$y_i = \gamma_i \, \hat{x_i} + \beta_i \tag{3.30}$$

During the inference phase, it is not desirable to estimate the mean and the variance of a given mini-batch, as outputs of the network should be deterministic and not dependent on other data points in the mini-batch. Once the network has been trained, the batch mean and variance in Equation (3.29) is therefore replaced by a global average over the whole training set or, more commonly, a running average that is computed during the training.

Another interpretation of BN is the fact that it can be seen as a specific type of regularization. Namely, during training the activations of each individual data point will now be dependent on the activations of the other data points (as the mean and variance are computed over all points). As mini-batches are shuffled during training this will lead to a regularization effect and it has been seen that BN in deep networks shares the same traits of regularization (Luo, Wang, Shao, & Peng, 2018).

Note, experimental studies have recently shown (Santurkar, Tsipras, Ilyas, & Madry, 2018) that it is likely that the effectiveness of BN does not come from the reduction of internal covariate shift, but indeed leads to a smoother optimization landscape and thereby more stable gradient behavior and faster training.

3.6 Convolutional neural networks

In Subsection 3.4, the main idea of feed-forward networks is shown. A Convolutional Neural Network (CNN) is a particular type of feed-forward neural network. While multilayer perceptrons make use of matrix multiplications (each input is weight by an independent weight), convolutional neural networks exploit the grid-structure that is inherent in data types such as images, videos and text. In the following sections we will introduce the essential components of CNNs.

3.6.1 Convolution

Convolution is the fundamental operation found in CNNs and is a widely used mathematical operation that expresses the overlap of two functions w and x as w is shifted over x. This can be defined as

$$s(t) = (x * w)(t) = \int x(a)w(t-a)\,da \tag{3.31}$$

and for the discrete case

$$s[t] = (x * w)[t] = \sum_{a=-\infty}^{\infty} x[a]w[t-a]\,. \tag{3.32}$$

In image processing our signals (or functions) generally are represented in two dimensions and the discrete convolution in that case is

$$s[i,j] = (I * K)[i,j] = \sum_{n} \sum mI[m,n]K[i-m, j-n]\,, \tag{3.33}$$

where:

I is commonly referred to as the input and K as the kernel respectively.

Convolution-based techniques are a standard tool in image processing. A filter matrix (kernel) is tuned to detect the presence of particular lines in the image results in the filtered image, as shown in Figure 3.13. Such images are in the context of CNNs, often referred to as feature maps (Wickstrøm, Kampffmeyer, & Jenssen, 2018). Figure 3.13 shows an example where a simple edge detection filter is applied.

An example of an edge detection filter is the well-known 3×3 Sobel operator filter matrix

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

The convolution operation slides the filter over the image one step at a time and computes the output for the current position as the weighted sum of the image pixels (that the filter is overlapping) weighted by the corresponding filter values. This exploits the idea, that a filter which can be used to detect for example edges in one part of the image can also detect edges in another part of the image. For instance, exploiting this structure of the image allows the edge detection filter of size 3×3 above to detect all the edges in the image by only making use of 9 weight parameters. Note, these weight parameters, the parameters of the filters, will be learned as part of the training as will be detailed below. To make this more clear, take for instance a 50×50 image. Applying a 3×3 filter and padding the image with one row/column of zeros on each side, we will obtain a 50×50 output image that contains all edges. If we would instead use a fully connected layer to do this, we would have to have $(50 \times 50)^2$ weights. This example illustrates why convolutions are preferable instead of fully connected layers when processing grid-like structure where you can assume that identical filters can be applied in different positions of the image (or video or text) and can be extract information from small local regions.

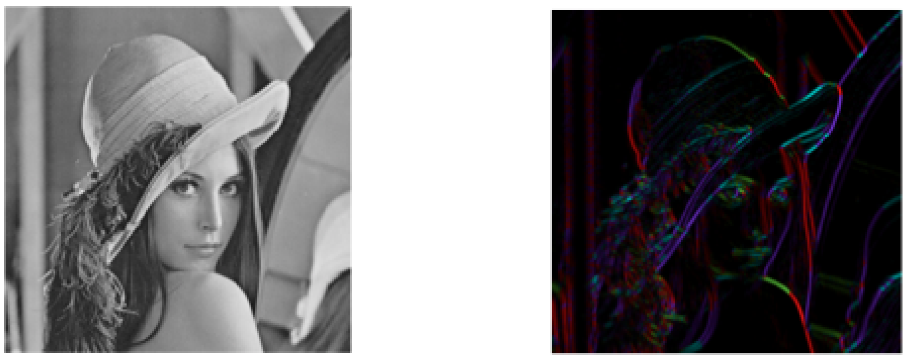


Figure 3.13: The original image to the left is convolved with a $[3 \times 3]$ edge detection filter resulting in the example image to the right. Images obtained from Edge Detection CSE (Gonzalez & Woods, 2006)

3.6.2 Convolution layers

In a convolutional layer multiple convolutions are applied to the layer input in parallel. This means, that given the input the layer might compute the horizontal edges, the vertical edges, and a smoothed representation based on three 3x3 filters. The output of these convolutions is then combined (along the depth dimension) and can be given as the input to another convolution layer. This means that the network can through a hierarchy of filtering operations obtain a representation that then can be used to perform a given task, such as supervised classification. In practice, however, similar to a fully connected layer, the filter weights are learned during training to find the ideal filtering operations that allow the network to perform best on a given task.

The size of the filter mask in neural networks is essential to define the connectivity relationship between the neuron weights in subsequent layers. The property which is called weight sharing means that all neurons of a feature map share the same weight matrix. Weight sharing reducing the number of parameters considerably since only one detector (e.g., the edge detector filter matrix) needs to be learned by convolution to detect objects all over the image. Since the same convolutions apply for the entire input image, it also meets the property of shift-invariance. A further advantage is that the convolution process does not depend on image size.

These advances allow CNNs reducing the amount of free parameter dramatically. CNNs make more extensive networks easier to train while at the same time maintaining the robust regularization effect.

3.6.3 Pooling

Pooling layers are a basic and central component of CNN, and often referred to as down-sampling layers. Typically this pooling operation reduces the data in the feature space by sub-sequentially down-sample the activations in small predefined spatial neighborhoods to a single value. Intuitively this reduces the feature space by mapping neurons within a region onto a single neuron. Pooling operations also addresses the characteristic of detecting a shape or a feature independently of where it is in the image, this is often referred to as translation invariance. The pooling layer merges activation values and operates upon each feature map separately and creates a new set of the same number of pooled feature maps each with a reduced spatial resolution. The total number of feature maps will not be affected by the merge operation.

Commonly, pooling operates on a 2×2 pixel grid or 3×3 pixel grid and reduces it to a single value by a mathematical operation such as the max or average. One new pixel thereby represents either 4 pixels or 9 pixels in the previous layer and assuming that the pooling is done without overlapping grids, it will lead to a reduction by a factor of 2 or 3 in each feature dimension. An example of pooling is illustrated in Figure 3.14.

Frequently applied pooling methods are max and average-pooling. Stride refers to the number of pixel-steps between two subsequent pooling grids. Another more recent method is the stochastic pooling, which serves as a regularizer method similar to dropout (Zeiler & Fergus, 2013).

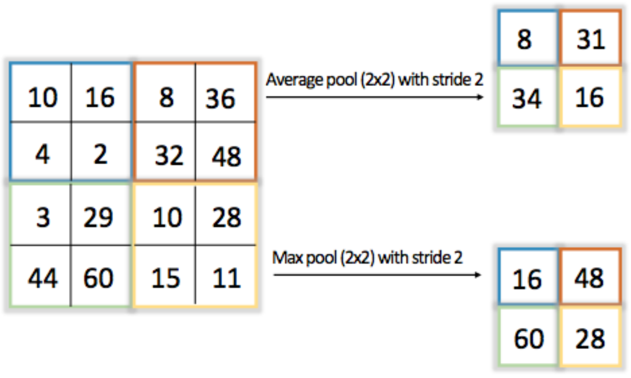

10
16
8
36
4
2
32
48
3
29
10
28
44
60
15
11
Average pool (2x2) with stride 2
8
31
34
16
Max pool (2x2) with stride 2
16
48
60
28

Figure 3.14: This example shows the effect of average and max pooling, and with a stride 2.

3.6.4 Architecture

A typical CNN architecture for classification tasks combines convolutional layers and pooling layers with fully connected (FC) layers similar to the one shown in Figure 3.15. After the convolution operations and the pooling, the fully connected layer aggregates all the information independent of spatial location and produces a feature vector for each image. This vector is then the input to the final layer which can be interpreted as a classifier on top of these extracted feature vectors. Activation functions or non-linearities are typically applied after the convolution operations. As previously discussed this activation function tends to be the ReLU function for deep networks (networks with many layers). In modern architectures, BN or alternative normalizations are further applied but are assumed to be included in the convolutional layers in Figure 3.15 for improved clarity (similar to the non-linearity).

3.6.5 Transpose convolution

Another operation that is commonly found in CNNs for segmentation is the transpose convolution layers. These layers can be used to learn an upsampling of a representation such as for instance bilinear upsampling, which is a central part of many segmentation architectures such as the FCN architec-

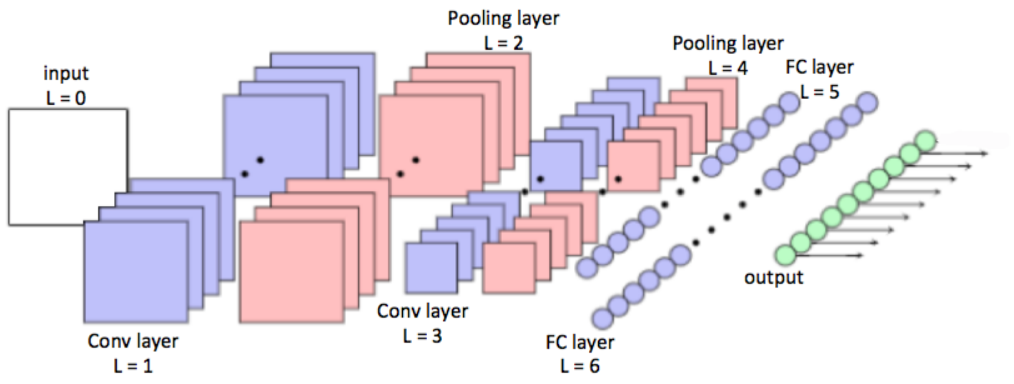


Figure 3.15: This example shows a typical illustration of the architecture of CNN. An alternating mix of convolution layers and pooling layers, where the pooling layers also incorporates a non-linearity and normalization stage. After the last pooling layer, the feature maps proceed through an arbitrary number of fully connected layers followed by an output layer that involves a softmax function typically.

ture (Long, Shelhamer, & Darrell, 2015), and the U-Net (Ronneberger et al., 2015) architecture, which will be discussed in more detail in Sections 3.7.

Transpose convolution is generally easy to implement by reverses the forward and backward passes of convolution. The image resolution improves by regulating the kernel size, padding, and stride of the operation. Assigning of a new pixel value is based on the weighted sum of nearby points and the weights of the kernel. Transpose convolution has the benefit that it can learn the weights for the up-sampling procedure, thus providing greater flexibility, but this adds more parameters to the network, which may lead to overfitting.

Transposed convolution is also referred to as deconvolution and fractionally stride convolution.

3.6.6 Data augmentation

Data augmentation is a technique we can use to reduce overfitting on models and to incorporate invariances by artificially extending the dataset. To perform well, a classifier needs a lot of training data such as images to train on (Perez & Wang, 2017). We can increase the amount of image training data by using the information only available in our image training data. A prevalent practice for augmenting image data is through simple techniques such as cropping, rotating, and flipping input images. An example of such an increase in image data is shown in Figure 3.16. Additional moves are color augmentations and further geometric steps, such as reflecting the image and changing the color palette of the image. All of the transformations maintain affine transformation methods of the original image. Previous work has demonstrated the success of data augmentation through simple techniques (Perez & Wang, 2017). However, the enlarged training samples are obtained from the original training data and they are not statistically independent and thereby do not have a comparable effect on gathering more real data (Wickstrøm et al., 2018).

The idea of data augmentation is not new, and in fact, various data augmentation techniques have applied to specific problems. The main methods fall under the category of data warping, which is an approach that seeks to directly augment the input data to the model in data space (Perez & Wang, 2017). The idea can be followed back to augmentations performed on the MNIST set in (Baird, 1992).

3.6.7 He Weight Initialization

Careless and randomly initializing of weights and bias can lead to difficulties, such as critical-high gradients of input signals. If there are too many out of bounds weights and biases, it may result in neurons getting saturated, and the gradients might vanish (Wickstrøm et al., 2018). The basic idea to address this problem of initialization is to investigate the variance of the responses in each layer. A response for a convolution layer is expressed as $\boldsymbol{y}_l = \boldsymbol{W}_l \boldsymbol{x}_l \boldsymbol{b}_l$, where we use l to index a layer, W is a d-by-n matrix, where d is the number of filters and each row of W represents the weights of a filter. Bias $\boldsymbol{b}$ is a vector, and y is the response at a pixel of the output map (He, Zhang, Ren, & Sun, 2015).

In (He et al., 2015) it was shown that an acceptable initialization method to avoid reducing or increasing the magnitudes of input signals exponentially results in the condition

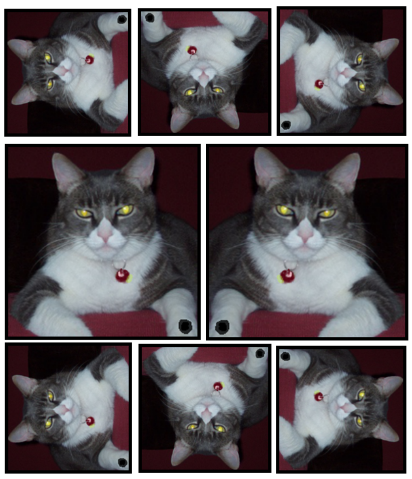


Figure 3.16: Shows input tabby cat image plus seven samples of images using data augmentation procedure. The input image is the left in the middle row, and the image to the right is the mirrored or flipped image. The six small images in the first and third rows are rotated samples from the images in the middle row (Note, disregard variation on image size) (Long et al., 2015).

$$\frac{1}{2} n_l Var[\boldsymbol{w}_l] = 1 \qquad \forall\, l. \tag{3.34}$$

that needs to be met, where l is the index of the current layer and assuming that weights are initialized with a mean of 0.

where:

n_l , refers to the the number of input weights, also referred to as "fan_in".

From their derivation they obtain the following condition

$$\frac{1}{2} n_{l+1} Var[\boldsymbol{w}_{l+1}] = 1 \qquad \forall\, l. \tag{3.35}$$

where:

n_{l+1}, refers to the the number of output weights, also referred to as "fan_out".

Given that we initialize from a zero-mean Gaussian distribution, this means that weights have to be initialized with $Var[\boldsymbol{w}_l] = 2/n_l = 2/fan\_in$ and $Var[\boldsymbol{w}_{l+1}] = 2/n_{l+1} = 2/fan\_out$, and also initialize $\boldsymbol{b} = 0$. Note, this can only be achieved when all layers have the same number of neurons and in order to make a compromise between the two conditions the average is commonly chosen such that

$$Var[\boldsymbol{w}] = \frac{2}{fan\_in + fan\_out} \tag{3.36}$$

3.7 Segmentation nets

Segmentation can be viewed as a pixel-wise classification task, where each pixel in the image is assigned to a given class. The first deep learning approaches to deep learning made use of this fact by employing the idea that a surrounding patch can be extracted from each pixel and can be used to classify the center pixel. This rephrases the segmentation task into a set of classification tasks, where the previously discussed CNNs for classification can be used. However, there are multiple issues with this naive approach. Firstly, in order to segment an image with HxW pixels, the task will be split up into HxW classification problems, leading to HxW forward passes during the inference phase. This is computationally expensive prohibitive, especially for large images. Secondly, classifying a patch with respect to its center pixel can lead to situations where a patch is supposed to get assigned to a specific class even though the majority (potentially all exept one) pixel belong to another class. The same issue appears for cases where a class boundary is in the center of the image and a shift of one pixel should lead to a completely different classification outcome. Due to this the naive approach usually achieves sub-par results and has been replaced with more modern approaches recently that are able to learn a segmentation mask for a given image in an end-to-end qualified learning procedure. Examples of these models will be the focus of the following sections.

3.7.1 Fully convolutional neural networks

The network architecture in Figure 3.17 was presented in 2015 by Jonathan Long, Evan Shelhamer and Trevor Darrel in the paper ”Fully Convolutional Networks for Semantic Segmentation” (Long et al., 2015). They presented the first end-to-end network architecture for pixel-wise prediction. Both learning and inference are performed for a whole image at a time, not requiring pre-and-post-processing.

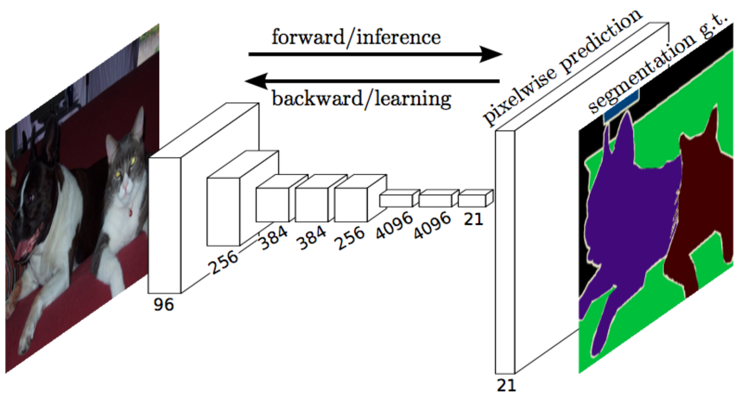


Figure 3.17: (Long et al., 2015). A Fully Convolutional Network (FCN) trains end-to-end by backpropagation. It is able to operate on an entire image as a whole, providing pixelwise predictions of the same size as the input image. Training is performed in a supervised manner by comparing the output predictions of the network with the ground truth segmentation mask.

A Fully Convolutional Network (FCN) requires densely (pixel-wise) labeled images for supervision. It operates on the whole image and learns to produce dense predictions in the form of a probability map. The training is performed in an end-to-end, pixels-to-pixels prediction manner. A benefit is the ability to adapt learned representation or weights trained from typical classification nets, such as LeNet (LeCun et al., 1989), AlexNet (Krizhevsky et al., 2012) and other subsequent networks in the encoder and only adds a decoding part that needs to be learned from scratch. This allows the use of networks that are trained on large classification datasets for weight initialization of the encoder. This process is commonly referred to as transfer learning and tends to improve performance for segmentation problems where limited ground truth images are available (due to the cost and effort required to provide dense ground truth).

In order to convert a classification network to an FCN, the fully connected layers (FC-layer) are replaced with convolutional layers. Note, for a fixed size image, a convolution that has a filter size that is equivalent to the size of the feature representation in a given layer can be viewed as (and is equivalent to) one neuron in a fully connected layer. This means, that adding the same amount of filters as there are hidden neurons in the fully connected layer will be equivalent and weights can be used interchangeable. The benefit comes from the fact that convolutions are independent of image size and the network can therefore be used to provide predictions for arbitrary-sized images after the modification. For larger images, unlike for the FC layers that produced only a single value for each neuron, the convolutional layers will now produce a representation that has a spatial extend for each filter. This representation can then be upsampled to the original image size providing pixel-wise labels. See the transformation in Figure 3.18. In order to perform the upsampling, one common approach is the use of so-called transpose convolutions (as discussed in 3.6.5). This method is applied in the FCN architectures (Long et al., 2015).

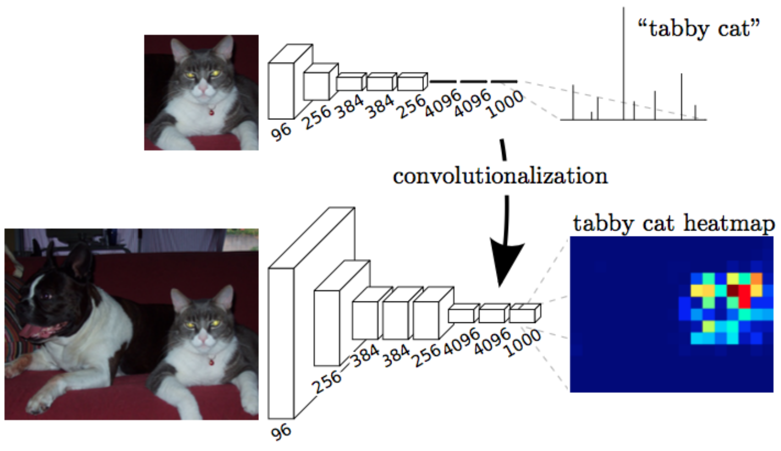


Figure 3.18: One can replace fully connected layers by convolutions to retain spatial information.

While pooling layers improve the field of view and thereby classification accuracy, it partially neglects spatial information. This is a drawback as it drastically reduces the spatial dimension of the input volume. This limit the accuracy of the segmentation (Harich, 2016). Typical recognition nets such as LeNet, AlexNet, and its deeper successors are classical kinds of networks architectures, designed to recognize visual patterns directly from images and commonly make use of multiple pooling operations leading to reduced spatial dimensions. For example, VGGNet consists of 5 pooling layers of stride 2 (Simonyan & Zisserman, 2014), result in a downgrade by a factor of 32 (2^5), and it may lead to inaccurate and coarser segmentation maps. To deal with this problem up-sampling in combination with so-called skip architecture are used in FCNs. Learning this network containing skip connections allows the system to recover the spatial resolution while preserving segmentation accuracy, see Figure 3.19

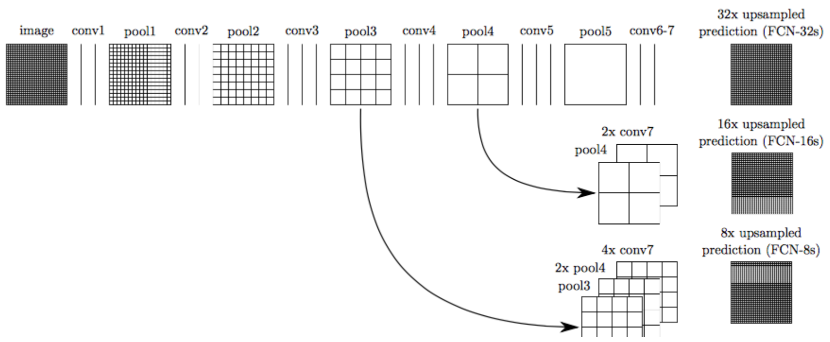


image
conv1
pool1
conv2
pool2
conv3
pool3
conv4
pool4
conv5
pool5
conv6-7
32x upsampled prediction (FCN-32s)
2x conv7
pool4
16x upsampled prediction (FCN-16s)
4x conv7
2x pool4
pool3
8x upsampled prediction (FCN-8s)

Figure 3.19: The figure illustrates the skip architecture nets, combining coarse high layer information with fine low layer information. To compact, the nets pooling and prediction layers showed as grids, and intermediate layers shown as vertical lines. The coarsest output is called FCN-32s on the first row and based on pool5 32 upsampled predictions only. The second-row net FCN-16s predict finer details, it combining the final stride 32 layers by a factor of 2 and the pool4. The third row the FCN-8s variant combines predictions of stride 32,16, and 8.

Figure 3.19 shows the architecture of the FCN-32s the most basic FCN-architecture as well as the FCN-16s and FCN-8s architectures that allow predictions with finer semantic details. The last two mentioned architectures were adding additional skip connections and thereby incorporating additional fine grade information at the cost of additional complexity. The first stages of the network are equivalent for all three structures and are based on the VGG-16 architecture (Simonyan & Zisserman, 2014), and each stage performs convolution followed by the ReLU (BN in FCN was not discovered at that point), then it is followed by a 2×2 max-pooling layer with stride equal to two. This process is repeated several times to produce a set of feature maps at decreasing resolution (Long et al., 2015).

The primary FCN-32s network output is directly up-sampled by a factor of 32 before passed into a softmax function. However, the 32-pixel stride in the final up-sampling layer limits the scale of detail in the final output prediction. Two upgraded FCN-version FCN-16s and FCN-8s address this issue providing a definite improvement by adding a novel "skip" architecture that combines deep, coarse, semantic information with shallow, fine, appearance information. For instance, FCN-16s provides an extra up-sampling. A link connects to the earlier layers with finer strides, first with a factor of 2 and then with a factor of 16. After the features from the coarse representation have been up-sampled by a factor 2, they are combined with the finer grained features and the combined representation is then up-sampled to the original image size. This is an intuitive approach, as it would be hard or impossible to produce an accurate pixel-wise prediction map for a 512×512 image only based on a 7×7 representation without adding higher resolution information from previous stages in the network. The FCN-8s version provides three up-sampling steps, two with a factor of 2 and then one final one with a factor of 8. See Figure 3.19 for details.

3.7.2 U-Net

The U-Net presented by (Ronneberger et al., 2015) was build upon the previous mentioned FCN architecture (Long et al., 2015). The original FCN architecture was modified and extended with the aim to operate with very few training images and nevertheless produce more precise segmentation. It is obviously a natural architecture to consider in regards to segmentation of medical images such as endoscopy modalities.

The U-net architecture is shown in Figure 3.20 and contains two paths respectively, on the left side a contracting path, the encoder, and on the right side an expansive path, the decoder. In the original U-net formation the structure of the contracting path was set up as a typical CNN (Ronneberger et al., 2015), (Long et al., 2015). Usual it is a 3×3 convolution followed by a ReLU and in addition, a 2×2 max-pooling operation with stride 2 for down-sampling and propagating the maximum activation to the next feature map. Additionally, it reduces the height and the width of the feature map to the halves of the previous layer. This results in a gradually spatial contraction. The standard CNN network of the encoder stops here, and the encoder maps all features to a single output vector.

Now, the expansion path aims to create a high-resolution segmentation map in the second part of the architecture. Every step consists of processing the features with a sequence of 3x3 convolutions followed by an up-sampling of

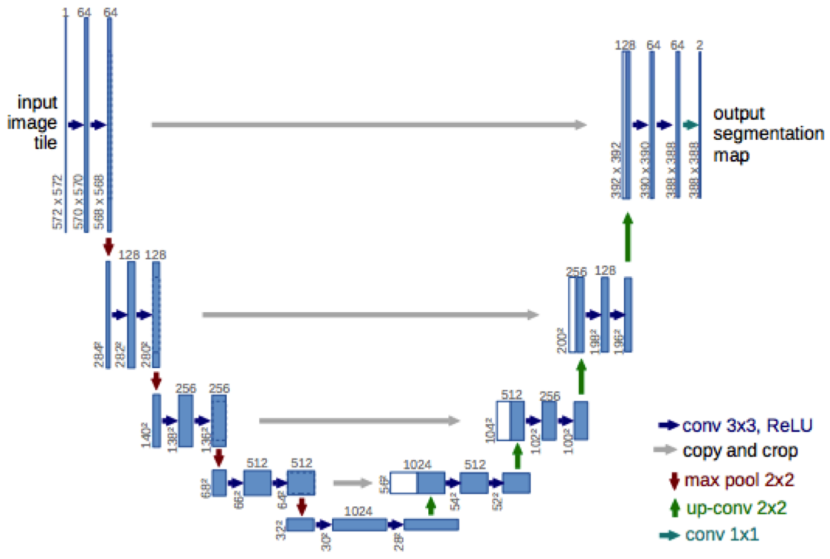


Figure 3.20: Architecture of original U-net.

the feature map by a factor 2 in both spatial dimensions using a learned kernel. In addition, every layer of the expansion path includes a connection to the corresponding layer of the contracting path to ensure an accurate recovery of spatial infomation via the skip connection. At the final layer a 1×1 convolution maps the feature vector into the desired number of output classes.

3.7.3 SegNet

SegNet is primarily an efficient architecture for pixel-wise semantic segmentation for a spatial understanding of typical road scenes such as buildings, road, and side-walk (*Badrinarayanan*, *Kendall*, *&Cipolla*,2015). The key learning module is a two-part encoder-decoder network. SegNet consists of encoder layers and corresponding decoder layers, followed by a final pixel-wise classification layer. There are five corresponding encoder and decoder layers, and hence it expresses a symmetric (but flipped) construction such that the decoder network is identical to the encoder network. In a given decoder layer, SegNet uses the max-pooling indices from the corresponding encoder stage to perform up-sampling, by placing activations into the corresponding positions. This produces a sparse feature activation map that is then processed by convolutional layers to make them dense. This approach, removes the need to explicitly learn the up-sampling and instead replaces it by additional "normal" convolutional layers. This architecture is shown in Figure 3.21.

The Encoder network of SegNet involves 13 convolution layers which correspond to the first 13 convolutional layers in the VGG16 network originating from the object classification (Simonyan & Zisserman, 2014). The fully connected layers of VGG16 are removed, making the SegNet encoder network significantly smaller with respect to the number of parameters, and thereby more comfortable to train compared to other recent architectures. Training weight from large classification datasets can still be used to initialize the network in a transfer learning manner. All individual encoder in the encoder network performs convolution with a filter bank to work out a set of feature maps. This is following by BN and then further by an element-wise tanh nonlinearity ReLU max(0, x). A max-pooling with a 2×2 window and stride 2 (non-overlapping window) and sub-sampling by a factor of 2 is applied to obtain the feature maps. The max-pooling operation will progressively reduce the spatial size of the feature map and achieve robust translation invariance over small spatial shifts in the input image (Badrinarayanan et al., 2015). Accordingly, several layers of max-pooling and sub-sampling can deliver more translation invariance and archive a significant image context for each pixel in the feature map, but also a progressive loss in spatial resolution of the feature maps, not beneficial for segmentation where exact boundary information is vital. Encoders must do a precise mapping of boundary localization before sub-sampling. This involves also memorizing of the locations of the maximum feature value in each pooling window and for each encoder.

In the encoder, the location of the maximum value (pooling index) of each of the 2x2 pooling window can be stored using 2 bits (per window) and can therefore be done very memory efficient. This avoids the need for learning the upsampling but does not address the loss in resolution caused by the pooling layer as it lacks the "skip" connections that include higher-resolution information that were found in the FCN. The decoding components use the max-pooling indices to up-sampled the features by placing them in the position that corresponds to the max-pooling location in the corresponding encoder. As this step provides sparse representations, the feature maps are convolved with a trainable decoder filter-bank (a set of convolutional layers) to make denser feature maps. The last decoder produces a feature map where the depth dimension corresponds to the number of K classes that the network needs to distinguish, which is then processed by a softmax layer. The softmax layer returns a K channel image of probabilities. The predicted

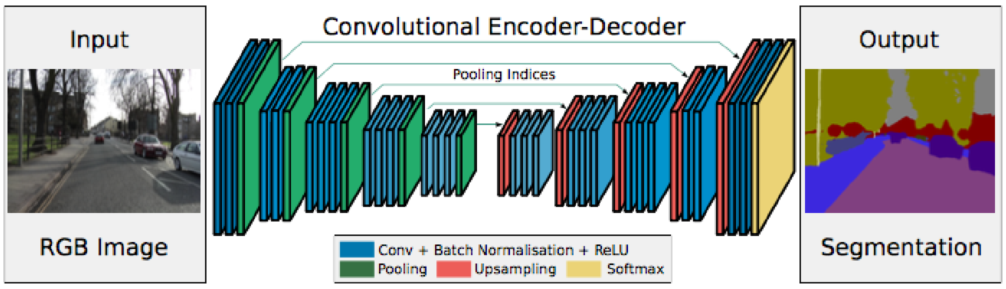


Figure 3.21: Shows an illustration of the SegNet architecture with the hierarchy of encoders and decoders. To create a bitmap, each encoder carries out dense convolution with a trainable filter bank. The appropriate decoder up-samples its input feature map(s) using the transferred pool indices from its encoder. Convolving these sparse feature maps with a trainable decoder filter bank produces dense feature maps followed by applying a batch-normalization step of each of these maps. After the final decoder, the output is fed to a softmax classifier which produces the final prediction for each pixel independently. The predictions can then be combined and displayed as a segmentation map (Badrinarayanan et al., 2015).

image segmentation then corresponds to the object(s) with maximum likelihood at each pixel. In contrast, other decoders in the network produce feature maps with the same size and channels as their corresponding encoder inputs (Badrinarayanan et al., 2015).

3.8 Class balancing

A major challenge when training CNNs for semantic segmentation is the presence of high class-imbalance in the dataset. Datasets are imbalanced when only a small number of training examples or pixels are represented in at least one class while other classes make up the majority of samples. In many classification tasks, the overall accuracy might be satisfactory and adequate, but class imbalance often represents an issue of ignoring small classes at the expense of the larger classes. When only considering the goal of archiving overall accuracy in a segmentation setting, small classes cover only a small area of an image and get less prioritized by the Net model in an effort to achieve the overall optimal score in the learning process (Kampffmeyer et al., 2016). The model wants to do as good as possible on as many pixels as possible, so then it will instead prioritize the classes that are very big at the expense of small classes. Low accuracy on the small classes will have no or minimal impact on the overall classification accuracy because of the much larger classes.

3.8.1 Median frequency balancing

Most machine learning algorithms do not work very well with imbalanced datasets where, e.g., pixels from one or two classes dominate the number of pixels among the classes in the training set. One can use different evaluation criteria and techniques for re-sampling the dataset, such as making a balanced dataset out of an imbalanced one. In segmentation tasks, one considers a dataset to be balanced when the number of pixels belonging to each class/segment is roughly the same. In most cases, the classes in the dataset are, however, not ever balanced. Further, since the class distribution in the individual images tends to be similar, removing individual images does not generally tend to address the problem and would also result in a reduced training dataset. For classification tasks, where labeling is less costly this is, however, a valid strategy to balance the dataset.

A major approach to overcome this problem is by boosting the importance of small classes. The idea is to assign small classes a higher cost value with the usage of a weighted cross-entropy loss function resulting in a more balanced classification result. By including the median frequency balancing (MFB) (Eigen & Fergus, 2015) which is, a weighted cross-entropy loss, the segmentation accuracy for small classes can be effectively improved (Bischke, Helber, Borth, & Dengel, 2018). MFB was first proposed by(Eigen & Fergus, 2015) to improve depth prediction and semantic labeling in imbalanced datasets.

In the segmentation task, the model receives the whole image as input and is tasked to generate a full resolution pixel-by-pixel labeled image. MFB weights the class loss by the ratio of the median for all classes in the training set, and the individual class frequency. The re-weighted cross-entropy loss function is as follows

$$L = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} l_c^{(n)} \log, \left(p_c^{(n)}\right)\omega_c \tag{3.37}$$

where:

N is the number of samples in a mini-batch.
$p_c^{(n)}$ is the softmax probability that the sample n is in class c.

The ω_c is given by

$$\omega_c = \frac{median\left(\{f_c \mid c \in C\}\right)}{f_c} \tag{3.38}$$

and expresses the class weight for class c where f_c is the frequency of pixels in the c class. C is the set of all classes.

Chapter 4

Deep learning for medical endoscopy

Deep Learning is one of the most popular trends in Machine learning research, and in particular CNNs have rapidly become a methodology of choice for analyzing medical images. In this chapter we will briefly present some of the prior work on designing automatic medical decision support systems for medical imaging with a particular focus on endoscopy.

4.1 Traditional machine learning

Historically, from early 2000's computational video analysis methods for automatic polyp detection have been developed first using flexible video endoscope, later also including WCE. At that time it was mainly integrated as a reviewing software from the manufacturer (Karkanis, Iakovidis, Maroulis, Magoulas, & Theofanous, 2000).

Several techniques to finding bleeding in the gastrointestinal tract are published in reviews and papers. This includes a variety of rule-based and traditional machine learning algorithm based methods are applied to extract color, texture and other features (Iakovidis & Koulaouzidis, 2015). Some of the papers describe how the use of the color space is applied in the detection of hemorrhage (bleeding), such as (Hwang, Oh, Cox, Tang, & Tibbals, 2006), where they proposed a technique to detect the bleeding regions automatically utilizing the Expectation Maximization (EM) clustering algorithm. Another paper (Karargyris & Bourbakis, 2008) proposed a novel methodology based on synergistic integration of methods, such as Color K-L transformation, fuzzy region segmentation, and Local-Global graphs for automatically detecting blood-based abnormalities in WCE videos. Berens et.al. (Berens, Fisher, et al., 2008) expanded their previous work with a new WCE video segmentation algorithm based on the hidden Markov model (HMM). They combined texture and motion features in addition to the color features from their previous work and improved the classifier results.

Still, many more individual works on traditional machine learning could have been mentioned, however, the interested reader is referred to a survey by Karargyris et.al. (Karargyris & Bourbakis, 2010) where they have reviewed several WCE and conventional flexible endoscopy videos. It shows a variety of WCE challenges, and that nearly all proposed methodologies by that time applied traditional machine learning to adapt to the tasks they carried out. All presentations were based on WCE and conventional endoscopy videos.

4.2 Deep Learning for medical imaging

Deep learning (DL) has made significant progress and demonstrated breakthrough performance over conventional machine learning methods across several fields (Ching et al., 2018). It is becoming largely used in the domain of medical imaging, especially some domain that need imaging data analyses, such as radiology (Hosny, Parmar, Quackenbush, Schwartz, & Aerts, 2018), where deep learning has among others been used for prediction of cardiology (Dong et al., 2018), ultrasound, where deep learning approach is used for detecting of thyroid papillary cancer in ultrasound images (H. Li et al., 2018), and dermatology, where they illustrate the classification of skin lesions using CNN (Esteva et al., 2017). The emergence of DL-based approaches aims to meet the desire of healthcare professionals to improve efficiency of the clinical work.

When it comes to the analyze of WCE images, deep learning has recently demonstrated promising results for bleeding detection. A novel bleeding detection approach based on an eight-layer CNN was proposed by (Jia & Meng, 2016). This network performs detection of both active and inactive bleeding frames achieving high-level accuracy scores.

4.3 Segmentation methods of angiodysplasia

For the detection and localizing of AD lesions various approaches have been proposed recently. Vieira et.al. (Vieira et al., 2016) presents a methodology to segment the ADs using different color spaces and a Maximum a Posteriori (MAP) method to divide the WCE image into two regions, based on spatial information using Markov Random Fields (MRF) theory. The segmentation was performed using a statistical classification based on Bayes rule where the posterior probability of each class was calculated. Another more recent method was proposed by (Shvets et al., 2018) and is closely related to our method and a summary of it is presented in Subsection 4.3.1.

4.3.1 Segmentation of angiodysplasia - clarification

Here, a review is provided of Shvets et al., 2018 for completeness. However, note that the objective of the work in this thesis is not on achieving overall best accuracy, but instead aims to analyze the effect of weighting the terms in the loss function and the effect of class imbalance. For the ease of training, experiments in this work are therefore done on a shallow FCN network that can be trained rapidly and overall accuracy will not be comparable to Shvedt et al., who use a much larger network that is much more time-consuming to train.

To illustrate what separates these works, a more detailed descriptions of the the differences is provided.

- To improve model generalization during training of the data set Shvedt et al. applied random affine transformations and color augmentation in HSV space.

- An optimal threshold was chosen for the trained model based on the validation dataset. Balancing this parameter can be seen as another approach of addressing the class imbalance by reducing false positives (fp) or false negatives (fn). In order to avoid this additional variable, this parameter is fixed to 0.5 in all our experiments. Choosing 0.5 is intuitive as the output can be interpreted as the probability of a pixel belonging to the AD class.

- Shvets et al. focused on the best performing accuracy score by evaluated 4 different deep architectures for segmentation, while this work's focus is on the class-imbalance problem.

- Shvets et al., 2018 used a generalized loss function by combining the common cross-entropy loss function of the U-Net with the Jaccard index in order to be able to directly optimize the model according to their chosen evaluation metric. It was claimed to improve overall segmentation performance with a limited amount of data.

Part II

Angiodysplasia Segmentation Using CNNs

Chapter 5

Dataset and Experimental Setup

This chapter describes the experiments that were performed as part of this thesis in order to evaluate the hypothesis that is on applying DNNs for the detection and segmentation of angiodysplasias on WCE images. We first present the content of the dataset that was analysed as part of this study before providing implementation details and describing the training process of the proposed architecture.

5.1 Dataset

The dataset we are evaluating has 599 color images obtained from wireless capsule endoscopy and in addition, there are 599 ground truths to the images. The images are in JPG and PNG format, all images with 576 $\times$ 576 pixel resolution. The origins of the dataset is from the Endoscopic Vision Sub-challenge[1]

The dataset is split into two folders or parts composed of 299 images and labels with apparent AD in one folder and 300 images and 300 labels without any pathology in the second folder. Pixel-level annotations have been provided by human experts in form of ground truth images. The ground truth images consist of 576 $\times$ 576 pixel binary masks in JPG format, where white

[1]https://endovissub2017-giana.grand-challenge.org/Angiodysplasia-ETISDB/

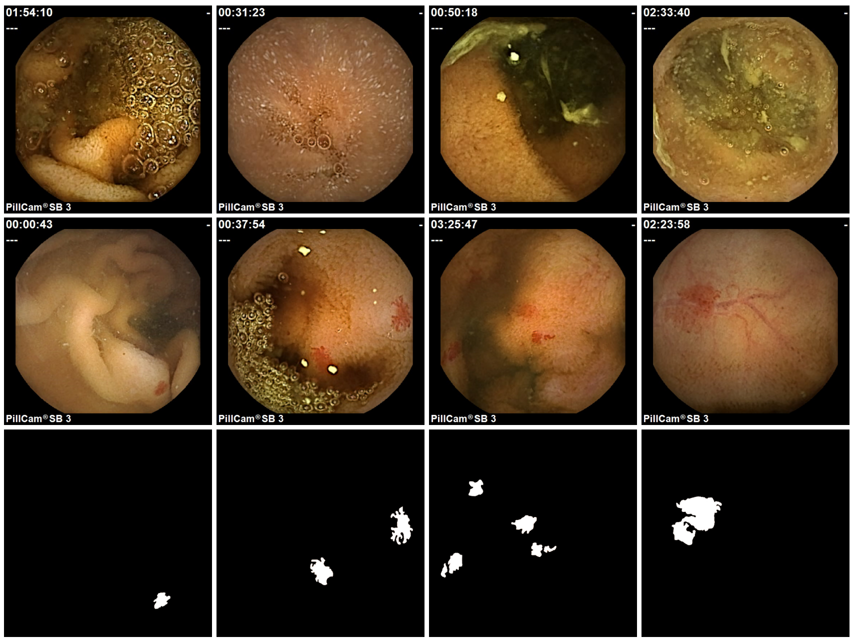


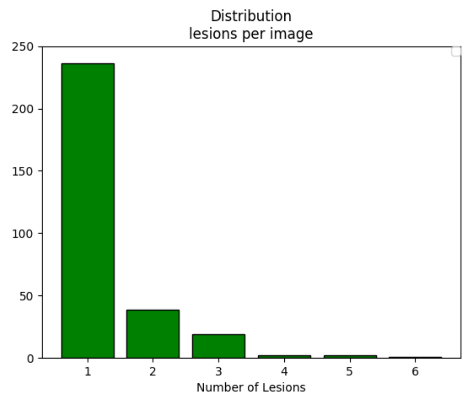
Figure 5.1: The top row corresponds to typical sample images without lesions. The middle row illustrates images with lesions and the bottom row contains the ground truth masks for the images in the middle row.

pixels correspond to lesions (angiodysplasias) located in the image and black pixels correspond to the background class.

Example images from the dataset are shown in Figure 5.1, where the four images in the first row are images without AD pathology. Images in the middle row have one to several AD lesions, and the last row contains labels that match the AD pathology we see in images from the second row. Figure 5.1 illustrates a problem in the dataset, namely the class imbalance, which is caused by the fact that lesions usually make up small objects in the images. Previous work has shown that class imbalance can be problematic for CNNs and we therefore analyse this class imbalance further in the following section.

5.1.1 Investigation of class imbalance

Figure 5.2 illustrates the previous imbalance further by displaying the overall AD lesion distribution for the dataset. In the left figure, we can observe that most images contain very few lesions. For instance 92% of the images have



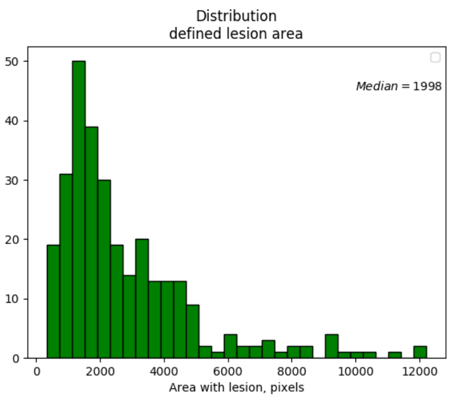


Figure 5.2: Distribution to the left show the number of lesions per image. The lesion area in the dataset is shown in the distribution to the right.

2 or fewer lesions and only one image has as many as 6 lesions. The chart to the right shows the distribution of pixels that corresponds to AD lesion per image. Again, we observe that most images contain very few lesion pixels. The images with the most lesion pixels contains around 12000 lesion pixels, which means that less than 4% of that image are covered by lesion. The median number of lesion pixels per image is 1998, which corresponds to around 0.6% of the image.

5.1.2 Dataset Preparation

In this subsection, we discuss how the dataset containing the AD-images was preprocessed and split into training, validation and test dataset and highlight the data augmentation techniques that were used to artificially expand the training dataset to improve model performance.

Cropping

In order to remove unwanted text annotations and irrelevant black pixels around the outside of the image, the original images (and ground truth annotations) were cropped from 576x576 pixels to the size of 480x480 pixels. To achieve this, 48 pixels were cropped on each side. For simplicity when regarding data augmentation (see below), we ensure that images keep the square shape after the cropping phase. The data augmentations will include image rotations and would result in unusual stretching of the objects if the height and width of the images differs and non-square images would include unwanted artifacts that would hamper the performance of the model.

Split

In a supervised learning model ground truth or labels are applied. The dataset is normally split into a training set, a validation set, and a test set. The training set is applied to train the model, while the validation set is applied for verifying the model and hyper-parameter tuning. Finally, the test set is applied to evaluate the performance of the model. This last step is done to evaluate the generalization performance of the model (Nielsen, 2015).

In order to have consistent data that gives reproducible results the actual dataset with 299 AD pathology images was randomized and divided into three sets. The training set contains 80% of the images and the validation and test set contain 10% of the data each. This results in a training set which contains 239 images and 30 validation and test images. Once the best hyper-parameters were found using the training and validation set, the model was trained and evaluated on the independent test set to analyse the predictive power of the model.

Augmentation

To improve deep learning performance an image classifier needs a fair amount of images to train on. When the training dataset only contain 239 images the size is quite limited. In addition to the previous pre-processing steps, there are further ways to improve the model performance by artificially increasing the training data size, namely data augmentation which was described in Subsection 3.6.6.

Augmentation has shown to be helpful to build upon as an improvement for network shortcomings (Perez & Wang, 2017). If the network model detects an object at some position but fails in situations where it has not been exposed to before, such as when the object is rotated, then it opens the ability to introduce data augmentation to artificially inflate the training set with rotated examples. We augmented the data by rotating the 239 training images in 90, 180, 270, and 360 degree interval, and repeated equivalently rotations using the same but mirrored image samples. This resulted in 8 rotated images out of just one and thereby expanded the training dataset from 239 images to 1912 images and augmented also the corresponding representation of labels from 239 labels to 1912 labels.

No augmentation was done on the validation set, meaning the dataset contains only original images.

Implementation Details

To introduce the datasets for DNNs there are different ways to do it here. We decided to adopt a Lighting Memory-mapped Database (LMDB). LMDB is an open-source software database of choice for large datasets. Four LMDBs were created from the original data, one containing the images of the training dataset, one containing the corresponding labels, one for the images of the validation dataset and finally one containing the labels for the validation dataset.

An alternative to the LMDB dataset, was to implement a tailor-made input layer[2] for the Caffe framework, however, over the course of this thesis it was concluded that the benefit of such a layer was limited compared to the LMDB approach and not proportional to the amount of work required to complete it.

The deep learning framework Caffe was used for the experimental setup and implementation of the models in CNNs. All evaluations were performed using the deep learning framework Caffe [3] on a single Geforce GTX 1080 (GPU). To support the segmentation of the WCE images and median frequency balancing the loss layer of the Caffe framework was modified (requiring modifications of the CUDA code to enable the code to run on the graphics processing unit (GPU)). The model specification mostly follows the specification in (Kampffmeyer et al., 2016) and will be described further in the next section.

[2]www.riptutorial.com/caffe/example/19019/prepare-image-dataset-for-image-classification-task

[3]https://caffe.berkeleyvision.org

5.2 Network

This section presents the network architecture of the model that was implemented as part of this thesis.

5.2.1 FCN Architectures

The FCN network was chosen as it represents one of the most widely used segmentation models. In order to be able to analyse the model quickly and avoid large training times, the FCN architecture was chosen to be reasonably light-weight. A summary of the FCN architecture is given in Figure 5.3. The architecture has a total of 9 convolutions and 3 max-pooling layers.

It has become commonplace to have a BN followed by each convolution stage (Long et al., 2015), unlike earlier FCN architectures that were non-normalize networks (Ioffe & Szegedy, 2015). The characteristics of BN is to allow for a higher learning rate (lr) due to smoother gradients and faster convergence to a minima.

Dropout is applied with a drop-rate of 0.2 and inserted after the last BN before the transpose convolution where the up-sampling is learned. Dropout improves the generalization performance of the network by reducing the problem of overfitting.

After the convolution layers and the pooling layers in the first part of the network, one convolution layer is used with a 1×1 kernel in order to take the 512 channel input down to a 2 channel input where each channel corresponds to one class (lesion vs background).

The up-sample layer is followed by a softmax function and succeeded by the weighted loss and a computation of the overall accuracy. Note that for the convolution operations, the notation "3×3 conv. 64" means that 64 different convolutions are performed, each having filter shapes of sizes 3×3. The depth of the filters is analogue to the number of channels to the convolution input.

Each layer of data in the FCN network is a three-dimensional array at size $h \times w \times d$, where h and w are spatial dimensions and d is the feature or channel dimension. The first layer takes in the RGB image, with pixel size $h \times w$, and d color channels.

The network has shown to be sufficiently large for satisfactory performance, while simultaneously being small enough to have a manageable memory capability and training time. Recall that the FCN-32s architecture has 16 convolutions layers and 5 max-pooling layers leading to a considerable increase in overall training time.

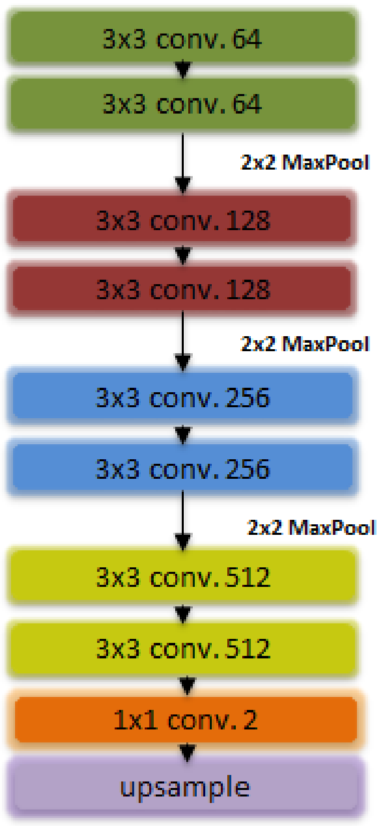


Figure 5.3: Architecture details for the FCN implementation.

5.2.2 Implementation details

Choosing the hyper-parameters can be somewhat challenging in DL applications since there is no straight forward way to find the best configuration for these parameters in such a high dimensional space. Hyper-parameter tuning is usually done by examining some portion of the hyper-parameter space by running several iterations of training and validation in combinations and then selecting the configuration which results in the best model performance. Further, hyper-parameter search can include the architecture, leading to the need to remove, change or insert layers. There is a whole field of research on finding the ideal architecture (Elsken, Metzen, & Hutter, 2019), with most approaches still being computational prohibitive. Unless something else is explicitly stated, the choice of the different hyper-parameters is based on monitoring the performance on the validation dataset during training.

Training stability

Both the FCN network and the FCN+weighting model were trained on batches that consisted of 12 images of size 480 × 480 pixels. The batch size was chosen due to the capacity of GPU memory. Note that the batch size in gradient descent is a tradeoff between the speed of each update step and the accuracy of the gradient estimation. Increasing batch size will results in a more stable and robust training procedure as several gradients are accumulated over the batch size and used to update the average loss of the loss function. However, each update step will be more costly due to the fact that less computation is required. Finally, (reasonably) small batch sizes have also shown to introduce stochasticity that can lead models to escape local minima (Montavon, Orr, & Müller, 2012).

Optimization Methods

There are several available optimization methods that build on the standard stochastic gradient descent procedure. However, the network was trained using the more recently proposed algorithm Adaptive Moment Estimation (ADAM) (Kingma & Ba, 2014). ADAM claims to be a little more robust in a wide range of non-convex optimization problems in the field of machine learning. In relation to the learning rate (lr), it adapts scale for different weights instead of hand-picking manually a single learning rate as is done in SGD (Kingma & Ba, 2014). Note, this is a common trade of most of the proposed optimizers as they remove the time-consuming task of fine-tuning the learning rate manually and make training more robust to the starting learning rate.

Other parameters

- Gamma: The learning rate for the model was dynamically reduced after each 2500 iterations (batches) by a factor of 10.
- Learning rate (lr): The training started with lr of $1 * 10^{-5}$ and was gradually reduced throughout the training due to the gamma multiplier.
- Optimizer: Networks were trained using the ADAM optimizer with default parameters, ($\beta_1 = 0.9$ and $\beta_2 = 0.999$) (Kingma & Ba, 2014).
- Weight_decay: Is a regularization technique also known as the L2-regularization (Nielsen, 2015). During training, the network weight decay was at 0.0005, see also Subsection 3.5.
- Class balancing: To address the high imbalance in WCE image datasets the FCN-network was trained with class weighting for the white pixel minority class at 0.9 and class weighting at 0.1 for the black pixel majority class. The reasons for this choice is further discussed in Subsection 5.2.3

5.2.3 Class weighting

In order to compute the class weights, we have to investigate the frequency of the lesion class and the background class. As we have two classes, the MFB weighting would consist of the mean of the frequencies divided by the class specific frequency and thereby in the following weights:

By iterating over all the images in the training set and counting the number of background and lesion pixels we get

Pixel class 0 = 54483717 pixel, approx 99%

Pixel class 1 = 581883 pixel, approx 1%

This results in the following weights:

$\frac{1}{2}(54483717 + 581883)/54483717 = 0.5$

$\frac{1}{2}(54483717 + 581883)/581883 = 47{,}3$

The difference would thus be at a order of 100x.

Empirically, experiments showed that such a large difference resulted in the model learning to over-segment the lesion class and the weights, due to the huge class imbalance, were therefore reduced to a factor of 10.

To result in a more balanced classification result, we assigned the FCN network a scaled cost value with the usage of a weighted cross-entropy loss function during training. The FCN network was trained several times on both methods where different weighting values were evaluated, but the highest accuracy for both classes was obtained by weighting the foreground class 0.9 and the background class 0.1.

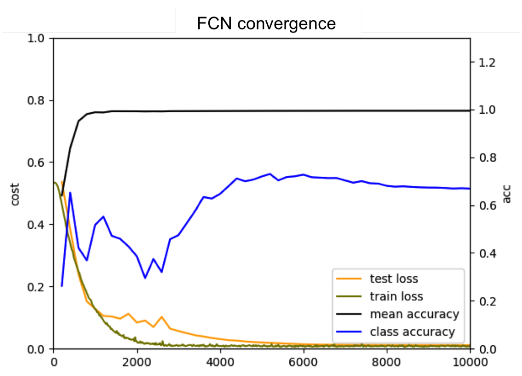
5.3 Analyses of the hyperparameters

This section summarizes and discusses the training process of the neural network architectures above. The two separate FCN-architectures are trained in an end-to-end manner on the WCE training dataset, one with and one without the class-weighted entropy loss, and the validation dataset is used to perform early-stopping of the training.

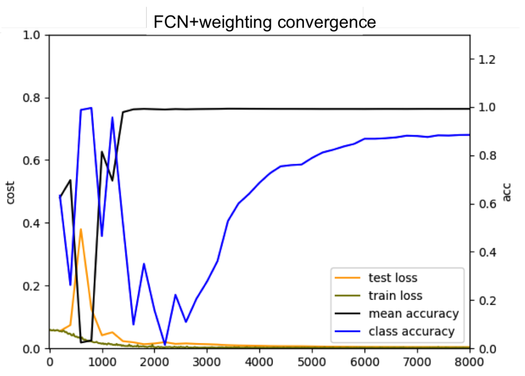
We further present an analyses of the effect of the weighting by presenting several additional results for different learning rates and different weightings.

5.3.1 Overall result

The learning curves during training for the best-performing models are shown in Figure 5.4. The model that achieved the highest validation accuracy was reached after 5000 iterations (snapshot) for the FCN model and 7500 iterations for the FCN-weighted model and training took approximately 15,5 and 23,75 minutes, respectively.



(a)



(b)

Figure 5.4: Learning curves: Shows the FCN accuracy on the training and validation dataset for the (a) and FCN+weighting accuracy on the training and validation dataset for the (b). For both figures, the two curves with blue color show the mean accuracy for the foreground class, and the two curves in black show the background class. The green and yellow colors are the loss curves for the train and validation loss respectively.

The accuracy on the validation set during training and the quantitative results obtained are shown in Table 5.1. The mean accuracy of the background class does not differ noticeably on any of the training methods and this is also the case for the total accuracies on both cases. The accuracy for the foreground class on validation was at 73.2% and 88.6% for FCN and FCN+weighting respectively and it shows an improvement on accuracy for the small foreground class when the class weighting is included in the training. The result was achieved with weighting 0.9/0.1 in favour the foreground class.

method	weighting foregr./backgr.	iter.	lr.	foregr. accuracy	backgr. accuracy	total accuracy
FCN		5000	$1 \cdot 10^{-5}$	0.732	**0.997**	**0.993**
FCN+w	0.9/0.1	7500	$1 \cdot 10^{-5}$	**0.886**	0.993	0.992

Table 5.1: Comparison of the best metrics for the FCN and FCN+weighting for the validation dataset.

5.3.2 Analyses of weighting scheme

In this section, the effect of weighting the classes in the loss function are analyzed. Results for different weightings are shown in Table 5.2 and illustrates that the overall accuracies are improving as the weighting approaches the setting 0.9/0.1 in favor of the foreground class. Rows four and five show that the foreground pixel accuracy decreases as the class weighting in favor of the foreground class is decreased. Increasing the weighting as shown in row one and two, does not further increase the foreground class accuracy, which suggests, that the model is unable to further increase on the foreground class accuracy and potentially is overfitting on the training dataset for the case where the weighting is 0.96/0.04. The third row shows the run for the chosen weight configuration. Recall that the foreground pixels class matches less than 1% of the total number of pixels.

method	weighting foregr./backgr.	iter	lr	foregr. accuracy	backgr. accuracy	total accuracy
FCN+w	0.96/0.04	12600	$1 \cdot 10^{-5}$	0.508	0.997	0.992
FCN+w	0.95/0.05	1400	$1 \cdot 10^{-5}$	0.884	0.840	0.841
FCN+w	0.9/0.1	7500	$1 \cdot 10^{-5}$	0.886	0.993	0.992
FCN+w	0.6/0.4	8000	$1 \cdot 10^{-5}$	0.562	0.996	0.991
FCN+w	0.5/0.5	13800	$1 \cdot 10^{-5}$	0.498	0.997	0.992

Table 5.2: Comparison the FCN+weighting metric results on the validation using various class weighting.

5.3.3 Analysis of early stopping

The accuracy results in Table 5.3 illustrate the effect of early stopping. Results for a single run are shown, when performing early stopping (at iteration 1400) and when continuing to train the model (to iteration 13800). The low overall performance for the model that was produced by early stopping (84.1%) is a bit misleading as it is due to the low background accuracy. However, the accuracy for the foreground class is considerably higher than for the model that was not stopped using early stopping. This is intuitive as the loss function, which is used to perform early stopping, is weight and will therefore prioritize foreground class accuracy over background class accuracy.

method	iter.	lr	foregr. accuracy	backgr. accuracy	total accuracy
FCN+weighting	1400	$1 \cdot 10^{-5}$	0.884	0.840	0.841
FCN+weighting	13800	$1 \cdot 10^{-5}$	0.498	0.997	0.992

Table 5.3: Comparison of metric results on the FCN weighting validation datasets, one with early stopping and one where the training continues.

5.3.4 Analysis of learning rate

The training examples in Table 5.4 illustrate different learning rate settings. The training in the first row was stopped early due to low accuracy and no (or very slow) improvement in the average loss. The third row shows the run for the chosen weight configuration. The training in the last row was instead stopped early due to the fact that the loss was increasing, due to the high learning rate.

method	weighting foregr./backgr.	iter	lr	foregr. accuracy	backgr. accuracy	total accuracy
FCN+w	0.95/0.05	2600	$1 \cdot 10^{-6}$	0.315	0.670	0.666
FCN+w	0.9/0.1	7500	$1 \cdot 10^{-5}$	0.886	0.993	0.992
FCN+w	0.95/0.05	400	$1 \cdot 10^{-4}$	0	1	0.989

Table 5.4: Results on FCN+weighting metric on the validation on various learning rate.

5.3.5 Summary of the analyses

In this section, several experiments were performed with many different hyper-parameters in order to analyze their effect on training. The results illustrate that the best performance can be obtained with a weighting of 0.9/0.1 in favor of the foreground class and a learning rate of $1 \cdot 10^{-5}$. Small changes in the parameter choices led to poor model performance, both with regards to convergence as well as resulting in considerably lower accuracy.

A full version of the FCN model, pre-trained on ImageNet, was also fine-tuned on the data at hand. However, it did not improve results.

5.4 Evaluations

The purpose of this Section is to analyze the result of weighting and describe the semantic segmentation accuracy of the experiments. The segmentation results are compared pixel-wise with the corresponding ground truth (GT) or labels and the semantic segmentation accuracy on the experiments follows the metrics described in (Fernandez-Moral et al., 2018). The authors describe three standard accuracy metric methods for segmentation evaluation which are

- pixel accuracy : $tn/(tn + fp) \quad and \quad tp/(tp + fn)$
- Intersection over union (IU/Jaccard) :
  $tn/(tn + fp + fn) \quad and \quad tp/(tp + fn + fp)$
- F1 Score/Dice : $2tn/(2tn + fp + fn) \quad and \quad 2tp/(2tp + fn + fp)$

Especially the F1-score and the Jaccard index, also called intersection over union (IoU), are typical metrics widely used to evaluate the classification results (Fernandez-Moral et al., 2018). However, all three metrics have various pros and cons.

The above metrics are computed in relation to the following confusion matrix in Table 5.5.

Since class frequencies are imbalanced and biased by the large dominant class it is most beneficial to evaluate the metrics per class, and further report the results of the average metric for the classes. Compared with global metrics, this class-wise average becomes usually less affected by imbalanced class frequencies (Fernandez-Moral et al., 2018).

	Predicted class	
True class	true positive (tp)	false negative (fn)
	false positive (fp)	true negative (tn)

Table 5.5: The Confusion matrix and notations. In the confusion square matrix each column represents the instances of a predicted class while each row represents the elements in an actual class.

The first metric, accuracy, evaluates the percent of pixels of that class that are classified correctly. On circumstances with imbalanced classes where one dominant pixel class dominates the image, accuracy is not an ideal metric. The next metric measures the IoU of the predicted labels of each class. The IoU considers both fp and fn and thus solves some of the major drawbacks considering imbalance described before and is a widely used metric for semantic segmentation.

The third measure the F1 score which is a function of two other metrics, namely

- Precision = tp / (tp + fp)
- Recall = tp / (tp + fn)

This F1 score might be a better measure to measure when seeking a balance between Precision and Recall. Precision is a good observation to discover when the costs of fp are high.

In general, the IoU metric tends to penalize single instances of bad classification significant more than the F1 score even when they can both agree that this one example is bad. Similarly to how L2 can penalize the largest spikes more than L1. So while the F1 metric measures something closer to average performance, the IoU metric reports a worst-case effect.

5.5 Results

The results on test set of the Deep learning segmentation on angiodysplasia lesions localized in WCE examinations are summarized and presented in this section. The quantitative results are treated first, followed by displaying the results of the segmentation.

5.5.1 Metric score

As discussed above, it is important to consider multiple metrics when evaluating models, especially in the presence of class imbalance. Table 5.6 summarizes the result of the experiments on the test set. The best scores are highlighted in bold. The pixel accuracy for the foreground class is highest for the FCN+weighting model, which agrees with our intuition as the weighting increases the foreground importance. However, this comes at a slight decrease in background accuracy. As previously seen, the overall accuracy will still be higher for the FCN without weighting as the background accuracy is slightly higher. Note, just by predicting a black background and disregarding all input leads to a global accuracy of about 99%.

	accuracy		IoU		F1 score	
model	foregr.	backgr.	foregr.	backgr.	foregr.	backgr.
FCN	0.683	**0.997**	**0.547**	**0.995**	**0.707**	**0.997**
FCN+weighting	**0.842**	0.994	0.508	0.992	0.674	0.996

Table 5.6: Metrics for the FCN and FCN+weighting for the test dataset.

From the IoU and F1 metric, we observe in Table 5.7, that the strong weighting that was used to train the model, tends to produce more fp for the FCN with weighting. This results in the lower performances on this metric.

	foreground pixel		background pixle	
model	tp	fp	tn.	fn.
FCN	44114	16092	6831359	20435
FCN+weighting	54342	42348	6805103	10207

Table 5.7: Reporting the pixel frequencies for the FCN and FCN+weighting models on the test dataset.

5.5.2 Underlying metric score

To further investigate the performance of the models, we look at the precision and the recall. From the precision score for the FCN model we can observe in Table 5.8 that because of the strong weighting, the precision score of the FCN model with the weighting was much smaller for the foreground pixel class, despite the large improvement of the tp frequency of the FCN+weighting model. Observation shows that it came at the expense of an increase of near three times for the fp pixel frequency, see Table 5.8. This means that even by improving the tp pixel frequency the expense is at the same time three times

model	precision score foregr.	backgr.	recall score foregr.	backgr.
FCN	**0.733**	0.997	0.683	**0.997**
FCN+weighting	0.562	**0.998**	**0.842**	0.994

Table 5.8: Underlying score for the FCN and FCN+weighting models on the test dataset.

as many of the fp pixels misclassified from the background class, leading to a worsened accuracy.

One interesting subsequent observation is that the recall score was calculated at 84.2% on the foreground class FCN+weighted model toward the FCN model score for the foreground class at 68.3%. The fn pixel frequency is just half of what it is on the FCN model, meaning a corresponding number of pixels are labeled as positive and added as the tp (the lesions class). This might indicate that the underlying structure of the FCN+weighting network output having more positive pixels within the region of detecting angiodysplasia disease. Additionally, the fp pixel frequency is three times as high, and for classification and segmentation of diseases such as angiodysplasia it can be argued as more preferable to present too many of the fp than the fn. This argumentation may be supported by inspecting the segmentation result images in Figure 5.5.

5.5.3 Result of segmentation

The qualitative result of the segmentation of six of the images from the test set are shown in Figure 5.5. The FCN algorithm identifies angiodysplasia lesions regularly but Figure 5.5 also illustrates the above mentioned problem of fp for the weighted model. This applies specifically to the image in the second and fourth column, where lesions are detected that are not in the ground truth. Similarly for the FCN model, we observe that it misses lesions (column six) as it aims to maximize performance on the background class. A good balancing of the classes is therefore crucial.

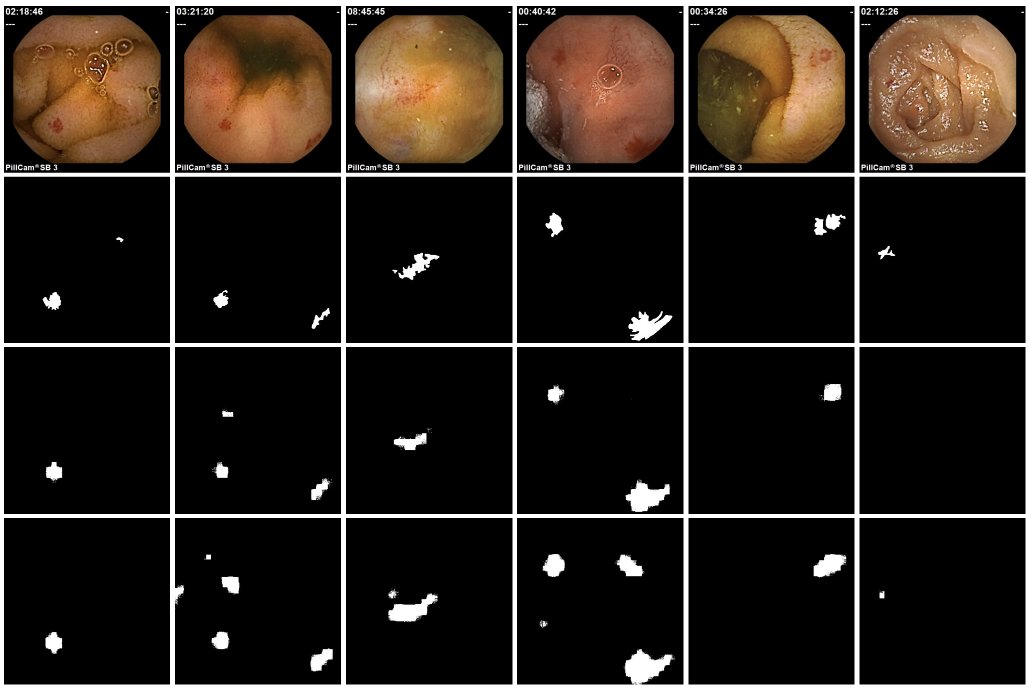


Figure 5.5: The two top rows show the images and their ground truth from the test dataset. The ground truth labels the angiodysplasia location output. The third row shows qualitative segmentation results using the FCN network, and the fourth row shows the qualitative segmentation results using the FCN network with weighting.

Chapter 6

Conclusion and future work

In this thesis, we analyze the impact of imbalanced classes for the task of angiodysplasia segmentation using convolutional neural networks. In order to address the high imbalance that is commonly found in the WCE image datasets, we utilize a class-balancing loss in order to increase the importance of finding lesions.

6.1 Future work

Results showed that a considerable amount of foreground pixel is getting misclassified by the models. Including weighting lead to an improvement in accuracy for the foreground class, but comes at a price as pixels of the background class get misclassified and performance according to metrics such as IoU and F1 diminishes. To improve overall accuracy, a larger and deeper network will need to be employed in future work.

As an alternative or extension of this work, the effect of adding the Jaccard index as done in Shvets et al. on the class imbalance problem can be analyzed as it directly incorporates tp, fp and fn.

Finally, a more optimal threshold can be obtained by using ROC curves to monitor how the four metrics tn, fp, fn, tp are changing when the threshold is moved.

6.2 Conclusion

In this thesis a deep neural network has been trained for the task of angiodysplasia segmentation from WCE images. Particular focus was put on the problem of class-imbalance, the problem that the important foreground pixels only make up a small fraction of the total pixels. Weights were introduced to the loss function in order to incorporate the importance of the different classes and the effect of these weights was analyzed.

References

Akbari, M., Mohrekesh, M., Nasr-Esfahani, E., Soroushmehr, S., Karimi, N., Samavi, S., & Najarian, K. (2018). Polyp segmentation in colonoscopy images using fully convolutional network. *arXiv preprint arXiv:1802.00368*.

Akhtar, A. J., Shaheen, M. A., & Zha, J. (2006). Organic colonic lesions in patients with irritable bowel syndrome (ibs). *Medical science monitor*, *12*(9), CR363–CR367.

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*.

Baird, H. S. (1992). Document image defect models. In *Structured document image analysis* (pp. 546–556). Springer.

Berens, J., Fisher, M., et al. (2008). Wireless capsule endoscopy color video segmentation. *IEEE Transactions on Medical Imaging*, *27*(12), 1769–1781.

Bernal, J., Sánchez, J., & Vilarino, F. (2012). Towards automatic polyp detection with a polyp appearance model. *Pattern Recognition*, *45*(9), 3166–3182.

Bischke, B., Helber, P., Borth, D., & Dengel, A. (2018). Segmentation of imbalanced classes in satellite imagery using adaptive uncertainty weighted class loss. In *Igarss 2018-2018 ieee international geoscience and remote sensing symposium* (pp. 6191–6194).

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., ... others (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Cheng, B., & Titterington, D. M. (1994). Neural networks: A review from a statistical perspective. *Statistical science*, 2–30.

Ching, T., Himmelstein, D. S., Beaulieu-Jones, B. K., Kalinin, A. A., Do, B. T., Way, G. P., ... others (2018). Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, *15*(141), 20170387.

Dong, N., Kampffmeyer, M., Liang, X., Wang, Z., Dai, W., & Xing, E. (2018). Unsupervised domain adaptation for automatic estimation of cardiothoracic ratio. In *International conference on medical image computing and computer-assisted intervention* (pp. 544–552).

Eigen, D., & Fergus, R. (2015). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture.

In *Proceedings of the ieee international conference on computer vision* (pp. 2650–2658).

Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. *Journal of Machine Learning Research*, *20*(55), 1-21. Retrieved from `http://jmlr.org/papers/v20/18-598.html`

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, *542*(7639), 115.

Fernandez-Moral, E., Martins, R., Wolf, D., & Rives, P. (2018). A new metric for evaluating semantic segmentation: leveraging global and contour accuracy. In *2018 ieee intelligent vehicles symposium (iv)* (pp. 1051–1056).

Gonzalez, R. C., & Woods, R. E. (2006). *Digital image processing (3rd edition).* Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Gurney, K. (1997). *An introduction to neural networks.* CRC press.

Harich, N. (2016). Fully convolutional networks for semantic segmentation from rgb-d images.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the ieee international conference on computer vision* (pp. 1026–1034).

Hinton, G. E. (2012). A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade* (pp. 599–619). Springer.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hosny, A., Parmar, C., Quackenbush, J., Schwartz, L. H., & Aerts, H. J. (2018). Artificial intelligence in radiology. *Nature Reviews Cancer*, 1.

Hosoe, N., Watanabe, K., Miyazaki, T., Shimatani, M., Wakamatsu, T., Okazaki, K., ... others (2016). Evaluation of performance of the omni mode for detecting video capsule endoscopy images: A multicenter randomized controlled trial. *Endoscopy international open*, *4*(08), E878–E882.

Hwang, S., Oh, J., Cox, J., Tang, S. J., & Tibbals, H. F. (2006). Blood detection in wireless capsule endoscopy using expectation maximization clustering. In *Medical imaging 2006: Image processing* (Vol. 6144, p. 61441P).

Iakovidis, D. K., & Koulaouzidis, A. (2015). Software for enhanced video capsule endoscopy: challenges for essential progress. *Nature Reviews Gastroenterology & Hepatology*, *12*(3), 172.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jia, X., & Meng, M. Q.-H. (2016). 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)* (pp. 639–642).

Kampffmeyer, M., Løkse, S., Bianchi, F. M., Livi, L., Salberg, A.-B., & Jenssen, R. (2019). Deep divergence-based approach to clustering. *Neural Networks*, *113*, 91–101.

Kampffmeyer, M., Salberg, A.-B., & Jenssen, R. (2016). Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In *Proceedings of the ieee conference on computer vision and pattern recognition workshops* (pp. 1–9).

Karagiannis, S., Goulas, S., Kosmadakis, G., Galanis, P., Arvanitis, D., Boletis, J., … Mavrogiannis, C. (2006). Wireless capsule endoscopy in the investigation of patients with chronic renal failure and obscure gastrointestinal bleeding (preliminary data). *World journal of gastroenterology: WJG*, *12*(32), 5182.

Karargyris, A., & Bourbakis, N. (2008). A methodology for detecting blood-based abnormalities in wireless capsule endoscopy videos. In *2008 8th ieee international conference on bioinformatics and bioengineering* (pp. 1–6).

Karargyris, A., & Bourbakis, N. (2010). Wireless capsule endoscopy and endoscopic imaging: A survey on various methodologies presented. *IEEE Engineering in medicine and biology magazine*, *29*(1), 72–83.

Karkanis, S. A., Iakovidis, D. K., Maroulis, D. E., Magoulas, G. D., & Theofanous, N. (2000). Tumor recognition in endoscopic video images using artificial neural network architectures. In *Proceedings of the 26th euromicro conference. euromicro 2000. informatics: Inventing the future* (Vol. 2, pp. 423–429).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Lecleire, S., Iwanicki-Caron, I., Di-Fiore, A., Elie, C., Alhameedi, R., Ramirez, S., … Antonietti, M. (2012). Yield and impact of emergency capsule enteroscopy in severe obscure-overt gastrointestinal bleeding. *Endoscopy*, *44*(04), 337–342.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521* (7553), 436.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1* (4), 541–551.

Li, H., Weng, J., Shi, Y., Gu, W., Mao, Y., Wang, Y., ... Zhang, J. (2018). An improved deep learning approach for detection of thyroid papillary cancer in ultrasound images. *Scientific reports*, *8* (1), 6600.

Li, Z., Carter, D., Eliakim, R., Zou, W., Wu, H., Liao, Z., ... others (2014). The current main types of capsule endoscopy. In *Handbook of capsule endoscopy* (pp. 5–45). Springer.

Liao, Z., Gao, R., Xu, C., & Li, Z.-S. (2010). Indications and detection, completion, and retention rates of small-bowel capsule endoscopy: a systematic review. *Gastrointestinal endoscopy*, *71* (2), 280–286.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3431–3440).

Luo, P., Wang, X., Shao, W., & Peng, Z. (2018). Towards understanding regularization in batch normalization.

Maieron, A., Hubner, D., Blaha, B., Deutsch, C., Schickmair, T., Ziachehabi, A., ... Schoefl, R. (2004). Multicenter retrospective evaluation of capsule endoscopy in clinical routine. *Endoscopy*, *36* (10), 864–868.

Manno, M., Barbera, C., Bertani, H., Manta, R., Mirante, V. G., Dabizzi, E., ... Conigliaro, R. (2012). Single balloon enteroscopy: technical aspects and clinical applications. *World journal of gastrointestinal endoscopy*, *4* (2), 28.

Montavon, G., Orr, G., & Müller, K.-R. (2012). *Neural networks: tricks of the trade* (Vol. 7700). springer.

Nadler, M., & Eliakim, R. (2014). The role of capsule endoscopy in acute gastrointestinal bleeding. *Therapeutic advances in gastroenterology*, *7* (2), 87–92.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (icml-10)* (pp. 807–814).

Ng, A. Y. (2004). Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on machine learning* (p. 78).

Nielsen, M. A. (2015). *Neural networks and deep learning* (Vol. 25). Determination press San Francisco, CA, USA:.

Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International conference on medical image computing and computer-assisted intervention* (pp. 234–241).

Rumelhart, D. E. (1986). Hinton, ge and williams, r. *Learning representations by backpropagation error. nature*, *323*, 533–536.

Sami, S., Al-Araji, S., & Ragunath, K. (2014). gastrointestinal angiodysplasia–pathogenesis, diagnosis and management. *Alimentary pharmacology & therapeutics*, *39*(1), 15–34.

Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2018). How does batch normalization help optimization? In *Advances in neural information processing systems* (pp. 2483–2493).

Sathiyabama, S. (2007). An efficient implementation of re sampling technique for the dynamic combination of multiple classifiers system.

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms.* Cambridge university press.

Shvets, A. A., Iglovikov, V. I., Rakhlin, A., & Kalinin, A. A. (2018). Angiodysplasia detection and localization using deep convolutional neural networks. In *2018 17th ieee international conference on machine learning and applications (icmla)* (pp. 612–617).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929–1958.

Theodoridis, S., & Koutroumbas, K. (n.d.). *Pattern recognition–fourth edition, 2009.* Academic Press.

Van de Bruaene, C., De Looze, D., & Hindryckx, P. (2015). Small bowel capsule endoscopy: Where are we after almost 15 years of use? *World journal of gastrointestinal endoscopy*, *7*(1), 13.

Vieira, P. M., Gonçalves, B., Gonçalves, C. R., & Lima, C. S. (2016). Segmentation of angiodysplasia lesions in wce images using a map approach with markov random fields. In *Engineering in medicine and biology society (embc), 2016 ieee 38th annual international conference of the* (pp. 1184–1187).

Wickstrøm, K., Kampffmeyer, M., & Jenssen, R. (2018). Uncertainty modeling and interpretability in convolutional neural networks for polyp segmentation. In *2018 ieee 28th international workshop on machine learning for signal processing (mlsp)* (pp. 1–6).

Xie, J., Girshick, R., & Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In *International conference on machine learning* (pp. 478–487).

Zeiler, M. D., & Fergus, R. (2013). Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*.