Using unsupervised machine learning as a tool for polyp detection in the GI tract

Mathias Kirkerod

Thesis submitted for the degree of
Master of science in Informatics: Technical and Scientific Applications
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2019

Using unsupervised machine learning as a tool for polyp detection in the GI tract

Mathias Kirkerod

Using unsupervised machine learning as a tool for polyp detection in the GI tract

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

Printed: Reprosentralen, University of Oslo

Abstract

Acknowledgements

my cat, if i had one

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

1.1.1 Introduction REM

Cancer is, today, the second leading cause of death in the world, only behind cardiovascular diseases.
It is one of the leading causes of mortality worldwide, with approximately 14 million new cases in 2012. It is defined as a disease that has an abnormal cell growth with the potential to spread into other parts of the body.Contrary to normal cells, cancer cells are often invasive, and it will spread if not treated. In contrast to many other diseases, cancer does not start from a foreign entity (such as a bacteria or virus), but it is often from a malfunctioning cell that starts dividing rapidly. This cell division can happen when a cell is damaged, by for instance by radiation or other factors specific proteins, or other chemicals. The result is that the cell either has damage in the DNA which contributes to abnormal cell division or the cell division itself malfunctions. In both cases the damage causes the cell to divide uncontrollably. Cancer can in some cases form without any external forces. The cell division is not always perfect, and dysfunctional cells might start a rapid division after being born. In most cases, this is not a problem, as most cells self destruct when they cannot operate.

Another factor that increases the risk of cancer is age. As we grow older, our body gets more prone to defective cell division and for each imperfect division the chance of getting cancerous cells increases. Our own body is designed to detect and remove cells that are prone to divide uncontrollably. Unfortunately, this system is not perfect, and the immune system can in some cases overlook cells that are cancerous. In either external or internal cases, cancer is by definition this uncontrollable multiplication.

Figure 1.1: Stages of cancer from wikipedia

1.1.2 Statistics on cancer REM

Since cancer is, in practice, just a cell division error, it has the opportunity to hit anyone, at any age. Because of this, it is a heavily reseached area, both in Norway, and the rest of the world. In 200X we spent 22Y million dollars worldwide on cancer research. Despite being such a researched area, it is still one of the top causes of human death. Some types of cancer, like cancer, is one of the simpler forms of cancer to treat, and at this point, those kind of cancers are non-fatal. *The most common types of cancer in males are lung cancer, prostate cancer, colorectal cancer and stomach cancer.***stewart2014world**

1.1.3 colorectal cancer REM

Humans can get cancer in every major organ, but some types of cancer are more common than others. For instance cancer in the gastrointestinal tract (GI) is one of the more common places to get cancer. Colorectal cancer is just behind x as the most common cancer for men, and it has a mortality rate of x in the first y years. We often call this 5-year survival rate for z. Z is the standard way to measure the life expectancy of a patient diagnosed with cancer.

1.1.4 polyps REM

Colorectal cancer often starts in polyps. Polyps are, polyps do.

1.1.5 preventative matters and early detection REM

-colonoscopy
-mri
-pillcam
As stated, the best way to fight cancer is to detect and remove it early, or in some cases, remove areas with a high chance of getting cancer. We classify cancer into four stages, and the stage the patient is in often determines the chance he/she has for survival. In general, the earlier doctors detect the cancerous parts, the more likely it is that the patient will survive. Moreover, as mentioned above, colorectal cancer often starts in these polyps. A crucial stage to prevent cancer lies in the early removal of there polyps. Reports show x about this

Because of this the ability to find, and remove colorectal polyps is excellent for preventing cancer in the GI tract.

colonoscopy/Ontonoscopy In the most common way to look for polyps in the GI tract is to use a medical team, and perform a colonoscopy or Ontonoscopy Colonoscopy is performed with a camera-stick that is inserted into the GI tract through the patients' anus.
Onoskopy is the same procedure; only the camera is inserted orally.

Advantages

- Accuracy: The use of a camera controlled by the doctor gives him/her the opportunity to stop at any anomalies.
- Quick results: Since the doctor is doing the procedure the result is given live.

Disadvantages

- Expensive: The cost of the doctor and the nurses needed is often high, especially on a routine check.
- Invasion of privacy: Getting a Colonoscopy or Onoskopy is a

MRI MRI (Maggnetic stuff) is the act of taking pictures blabla blabla
MRI (Maggnetic stuff) is the act of taking pictures blabla blabla
MRI (Maggnetic stuff) is the act of taking pictures blabla blabla
Advantages

- This is why MRI is good
- This is why MRI is good

Disadvantages

- This is why MRI is bad
- This is why MRI is bad

pillcam In the last 3-4 years, there have been testing and development on the pillcam project EIR. Machine learning has, through many of the earlier projects, got the detection rate for the polyps up to x%

Advantages

- This is why pillcam is good
- This is why pillcam is good

Disadvantages

- This is why cam is bad
- This is why pillcam is bad

1.1.6 Simulas contribution to the pillcam project REM

Simulas EIR
* CAD ACD (computer-aided diagnosis, Automated computer diagnosis)

Chapter 2

Background

In this chapter, we will present the background and motivation of our thesis. We start with our background in medical procedures, looking at how doctors perform colonoscopies, mainly from a gastrointestinal perspective. Then will then look at what the objective is for the medical staff, with different anomalies in the GI tract. Then our focus is moved to how doctors use computer-aided diagnosis (CAD) today to help with the screening. Lastly, we look at current models for CAD made both by Simula.

After the discussion from a medical point of view, we shift our focus to machine learning and give a brief introduction to different machine learning methods. We will look at how machines can ĺearnánd discuss different areas we can use and areas we are using machine learning today. We will then go in depth into the examples and look at the most common machine learning models. We will look at the most common form of machine learning, namely neural networks, and show the structure that is most used in this type of machine learning.

With this in mind, we will look at neural networks, especially convolutional neural networks, and how they work.

Lastly, we will combine the need for computer-aided diagnosis with the machine learning.

2.1 The Medical Background

In the field of medical diagnosis, there are always new and interesting methods being researched to help the medical staff when it comes to patient *rate of survival*, and quality of life. Everything from x to y is ways the medical industry has done to improve the survival rates of their patients. In the last decade *comuters and cameras came to help us.* Another example is the invention and usage

of gastro-stick-with-camera.

2.1.1 colonoscopy/gastro/procedure

When performin gastronomi we use astik

2.1.2 Medical images/data/other

2.1.3 Systems in place for detection

2.1.4 summary

2.2 Machine Learning

We have looked at the challenges that the medical staff has when it comes to detecting polyps, and how it is solved today. However, to truly understand how automated systems like Mirmirworks, we need to look at how Machine learning helps with the detection of the anomalies the medical staff are after.

Machine learning is a broad term, but can, in short, be summarised by:

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with the experience E. **MitchellTomM1997Ml**

A few things of note from this quote is the variables mentioned in the quote. Experience e is the stored knowledge the program has gotten. It is in most cases just numbers used to approximate a solution given an input, to try to get it as close to the right answer. This approximation is made for every task T until we are happy with the result. Lastly, to tell how well our program does we need a measure P that tells us how far away from the desired output we got.

From this, we see that the goal of machine learning is to improve some performance P with experience. This behaviour is a mimicry of human learning, where we as humans need to practice on a task to improve on it. As the amount of experience increase, both for us and the machine, the performance of the task becomes better and better. With this in mind, we can assume that, given the right amount and type of data, our machine learning algorithms can solve any problems humans can solve, given both the right training environment and a way to train, given the environment.

We have talked about machine learning in broad terms at this point. We have drawn a parallel between how humans learn, and how machines gather experience. Now we will look into the most popular machine learning techniques, and show the machine learning algorithms stores the experience gathered.

2.2.1 Machine learning types

With a basis in the quote from **MitchellTomM1997Ml** , we have a broad definition of what machine learning can be. As long as we have a model trying to complete a task based on previous experience, it can be called machine learning. Though just like humans, we have multiple ways to gather and retain information.

Figure 2.1 shows a chart over the three most common categories within the field of machine learning.

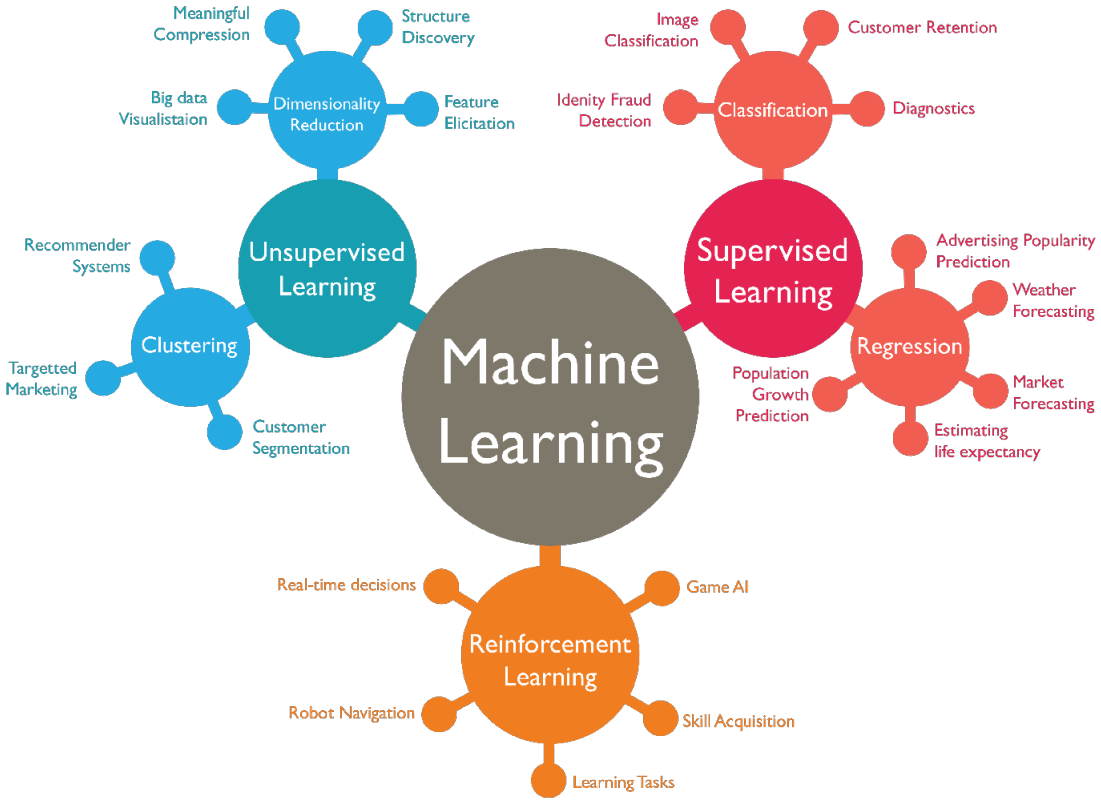

Figure 2.1: The three main types of machine learning and their subtypes

We have three subcategories of machine learning: Supervised, Unsupervised, and Reinforcement learning.

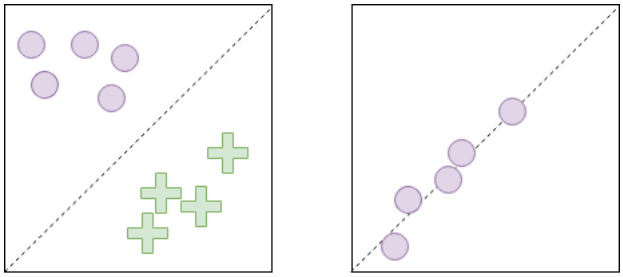

Figure 2.2: Left: Example of binary classification. Right: Example of regression

supervised machine learning

Supervised machine learning is the simplest form of machine learning to understand. It is also the most prominent where x% is and

Supervised machine learning functions have the objective of, given an input-output pair, approximate the input to be as close to the output. Alternatively, in simpler terms, given an input x, produce an answer as close as possible to the output y. A supervised algorithm analyses the training data and produces an inferred function, which can be used to map new data entries.

Supervised learning: is the act of training with data that has an answer or a label. The learning algorithm can get supervision while training on the task. An example of a supervised task is to recognise handwritten numbers, or differentiate between dogs and cats. The task is considered supervised if the images come with the correct label in the data set. These examples are typical classification examples, where the task is to identify the right group to classify the data to A simpler classification assignment is binary classification, where the target is (often) yes or no. Examples for binary classification is if an email is spam or not, is a car Norwegian or International. In the last example, the classification changes from binary to multi-class if we sort the cars on every nationality, and not just Norwegian/non-Norwegian. Another type of a supervised learning task is regression. Regression is the act of prediction given prior data. Examples of regression are everything from the prediction of stock prices, to house prices in an area, to

unsupervised machine learning

Unsupervised learning: is the act of training without any supervision, on the sense that we do not give the algorithm the answer to the training data set. Since we do not have categorised data in unsupervised learning, we often Types of unsupervised learning can, for instance, be clustering, the act of sorting data

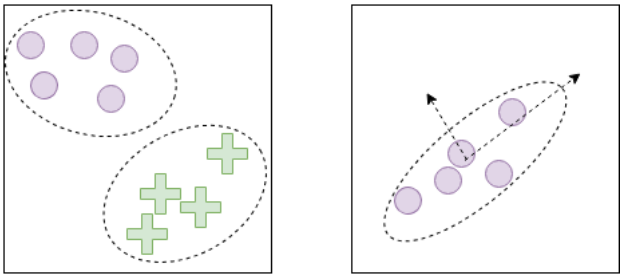

Figure 2.3: Left: Example of binary clustering. Right: Example of principal component analysis

based on similarity. An example of this can be if we want to sort plants based on species, or we are detecting anomalies in a dataset. Unsupervised learning can be used for PCA or other dimensionality reduction methods.
A third method to used unsupervised learning is the adversarial route, where we use machine learning to make similar looking data to the original data set.

In the description of supervised vs unsupervised we looked at a specific branch of machine learning: Classification. Classification is, as the name implies, the task of getting data sorted into groups of similarity.

- subsfication
- r to the pillcam projression
- transcription/translation
- de-noising /finding missing inputs

rienforcement learning

Famous machine learning types

Now that we have a basis on the three types of machine learning we can go into more detail on the most successful types of machine learning used both now and in the past. Machine learning was coined as a term as early as in the 10000 century. The first concept was of the

In more recent times, when the first computers came, a common type of machine learning was the K-nearest neighbour algorithm.

Another early adoption of machine learning was in the form of regression. Regression is the statistical concept of estimating the relationships among variables. It is in heavy use today, and one of the core concept we use machine

learning for today. Legendre first used regression in 1805 with his method of least squares.The least square methods were first done by hand, and it was at the time one best models, backed by math, to estimate the relationship between an input and a subsequent output.

Today regression analysis is widely used in statistics and informatics, and there is a significant overlap between the two research fields. While often we can make analytical models when working with a dataset with few variables, machine learning has the possibility of making much more complex models.

A newer and applied form of supervised machine learning is the support vector machine. The original support vector machine was invented by Vladimir N. Vapnik and Alexey Ya Chervonenkis in 1963. In 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik suggested ways to make the support vector machine work in multiclass examples by using kernels.

A support vector machine is a form of classifier where the goal is to divide two datasets by a line that is just between the data.

We have now discussed the general structure of the three types of machine learning. For each of the three methods we have looked at designs that utilise their form of learning, and we have shown how they can be used in the real world. We have also looked into some successful algorithms through the ages, highlighting innovations that helped form our vast library of methods we can use to tackle problems we meet. We will now first go more in-depth into how a general machine learning algorithm works, giving a rundown on how a simple algorithm works from start to end. After this, we look into a more advanced example utilising a Generative adversarial network as a template.

2.3 How machine learning works

One of the easiest to understand tasks in machine learning is the process of regression. As stated earlier, regression is a process of approximation given prior input.

We start with one of the simplest forms of approximation, namely linear approximation. In linear approximation, we are interested in finding the function that best defines our data using only a polynomial of the first degree. First, we recall that a first-order function is always on the form

$$y = ax + b \tag{2.1}$$

Where x is input, y is output and the constants a and b defines the function.

Figure **??** shows an ideal example of linear regression. Here we approximate the values of our model with the straight line defined by choosing the right

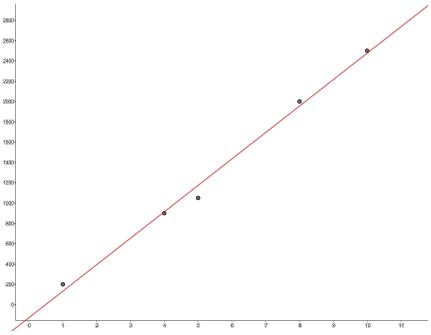

Figure 2.4: Example of linear regression in Geogebra. Here the red line is the best approxmiation of a y value, given an x value.

slope (a) and the right constant (b). With the knowledge of math, we look into how to do it computationally with the help of simple machine learning.

We can recall from the quote by **MitchellTomM1997Ml** that we gain experience E by doing a task T. In our example we choose to store our experience as its done in equation 2.2.

$$y = W^{(1)}x + W^{(2)} \tag{2.2}$$

Here, like before, our output is y and our input is x. We have replaced a and b with new placeholders $W^{(1)}$ & $W^{(2)}$. Now our goal is to, given a task T, gain experience E and store it in $W^{(1)}$ and $W^{(2)}$. With our values for $W^{(1)}$ & $W^{(2)}$ we want the best performance P. The best performance here is defined as getting the smallest difference between the predicted output data and the actual output data.

The most prominent way of calculating this error is to use the mean square error between the predicted and actual output of the data.

$$MSE = \frac{1}{2m}\sum_{i}(\hat{y} - y)_i^2 \tag{2.3}$$

Where m is the number of samples, y is the real output, and $\hat{y}$ is the predicted output. The 2 in the denominator is just a constant to make the derivation of the formula easier.

From this, we can intuitively see that the error tends towards 0 when $\hat{y}$=y. We can also note, because of the squaring in the formula, that the error is only based on L2 distance between $\hat{y}$ and y.

Now that we have an error, we need a way to improve it At this point, we have a way to store experience E (in $W^{(1)}$ & $W^{(2)}$), measure performance P (in the MSE), and we have tasks T (in the form of input-output pairs).

Given an input-output pair, we will now look at how to use machine learning to better approximate the next input-output pair.

Lets start with:

$$x = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} y = \begin{bmatrix} 1.5 \\ 2 \\ 2.5 \end{bmatrix} \tag{2.4}$$

with the weights $W^{(1)} = 0$ and $W^{(2)} = 0$ Using the formula 2.2 we can calculate $\hat{y}$ to be

$$\hat{y} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.5}$$

We can now calculate the performance by applying the mean square error function. Using 2.3 gives us a loss of:

$$\frac{1}{2*3}((1.5-0)^2 + (2-0)^2 + (2.5-0)^2) = 2.08 \tag{2.6}$$

With our new found error, we need a way to use this to update our weights $W^{(1)}$ & $W^{(2)}$ to get a better estimate.

The most common way to update our weights is to use gradient descent. Gradient descent is a first order iterative optimisation algorithm for finding the minimum of a function. In our case, we are looking for the minimum value of the MSE function. Gradient descent is defined as (simplified for our example):

$$a_{n+1} = a_n - \gamma \nabla F(a_n) \tag{2.7}$$

Where ∇F() is the derivate of function in question, a is the input at step n, and γ is a learning rate set to a small number.

Derivating 2.3 and using a learning rate of 1 we get the following.

$$\nabla F_{W^0} = \frac{d}{d_{W^0}} \frac{1}{2m} \sum_i (\hat{y} - y)_i^2 \nabla F_{W^0} = \frac{1}{m} \sum_i (\hat{y} - y)_i \tag{2.8}$$

$$\nabla F_{W^1} = \frac{d}{d_{W^1}} \frac{1}{2m} \sum_i (\hat{y} - y)_i^2 \nabla F_{W^1} = \frac{1}{m} \sum_i (\hat{y} - y)_i \dot{\hat{y}} \tag{2.9}$$

Feed forward

Loss and gradient decent

2.4 Neural Networks

We have looked at different types of machine learning, and we have gone in depth into how a linear regression model works. In this chapter, we want to get further insight into how we can make more complex models, and we will look into the most popular method for machine learning, namely Neural Networks. After the rundown on how neural networks are built up and how they operate, we will look into convolutional neural networks. In the end, we will look at successful networks, mainly made for image classification.

2.4.1 The perceptron

To explain how a neural net operates, we first need to look at the most fundamental structure present in every type of neural network, namely the Perceptron. The fist perceptron was introduced by in , and it was made as an attempt to mimic the human neuron.

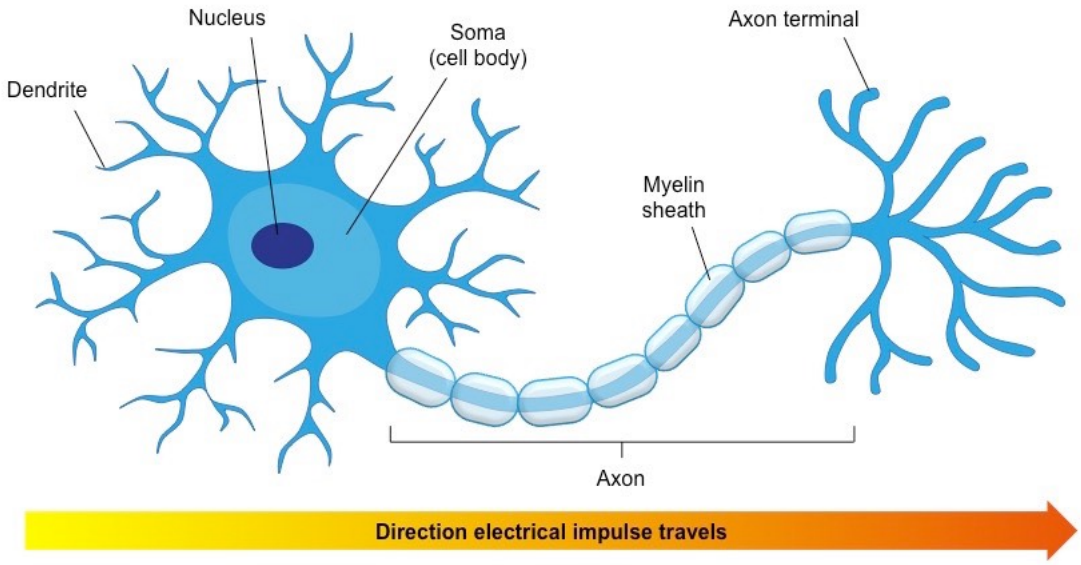

Figure 2.5: https://socratic.org/questions/how-is-a-neuron-adapted-to-perform-its-function

Figure 2.5 shows how a human neuron looks like, and in which direction the

signal goes. Each neuron is connected to multiple other neurons by connecting the dendrites to other neighbouring neurons. When a signal is sent, the dendrites register the signal and sends it through the axon out to the axon terminal. At the axon terminal, other neurons pick up the electrical signal and pass it through their axon. This flow of electricity is the fundamental way different part of our brain communicates, and the different pathways the signals can take represent how we learn.

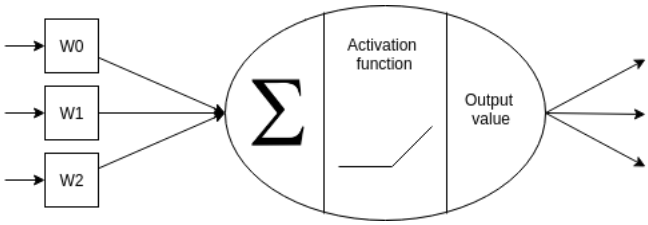

Figure 2.6: Simple perceptron

We can now look at the perceptron in figure 2.6, as we can discern, this mathematical model has the same structure, and we have the same flow as we would in the human brain. First, we get numerical data from, either other perceptrons, or from an input. We add it together and send it out on the other side. We will now look more in depth how this is done, and show how the mathematics behind it works.

Step 1: our perceptron gets signals $a_{(i,0)} - a_{(i,n)}$ where n is the number of inputs to the perceptron.

Step 2: each of our signals $a_{(i,0)} - a_{(i,n)}$ is multiplied by a weight $W^{(0)}$ - $W^{(n)}$.

Step 3: we sum every input to a scalar.

Step 4: We apply an activation function. This can be a simple sigmoid function or tanh function, but the most common is the Rectified linear unit (ReLu)

Step 5: The output of the activation function $a_{(j,0)} - a_{(j,n)}$ is sent to the next perceptron.

2.4.2 Multilayer perceptrons

The neural network was a proposal made by Warren McCulloch and Walter Pitts (1943) created a computational model for neural networks based on mathematics and algorithms called threshold logic.

The first multilayer network at the time used backpropagation in the same way described in .

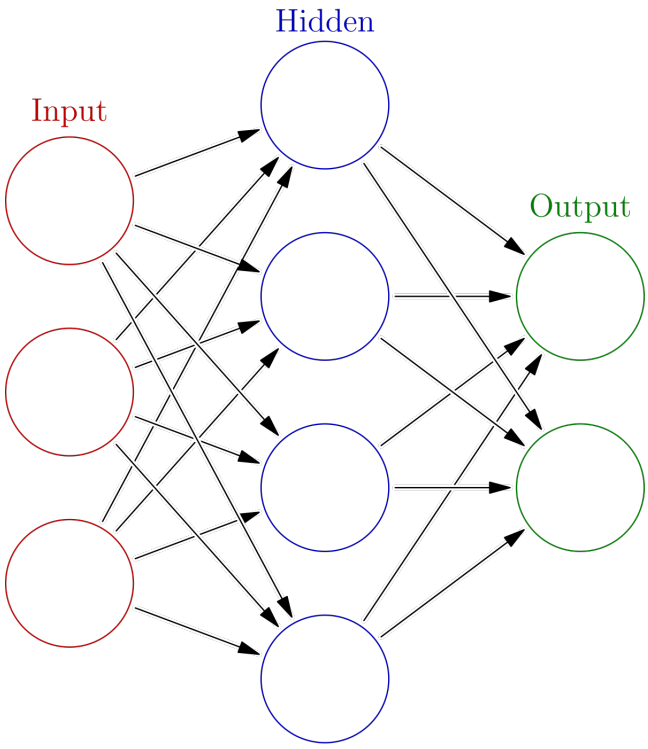

Figure 2.7: THIS IMAGE IS SHAME(LESS)LY taken from the internetz, draw own so the lawyers don't get you!

Figure 2.7 shows the basic structure of a multilayer network. In this figure, we have four things we need to look into.

2.4.3 Feed forward and backpropagation through a neural network

2.4.4 Convolutional neural networks

The multilayer perceptron we have discussed is a strong tool that can learn a multitude of decision boundaries, and subsequently learn to classify thousands of different classes. As we get more data and more classes, the networks needed to solve our problem grows. We can recall from chapter that the number of weights between neurons is . As the number of perceptrons per layer in our

neural networks increases, the total amount of storage space increases by a factor of 4

Another problem with the standard MLP is the fact that it is spatially dependent. Given an input x, the output, y, of the MLP will vary a lot if we shift the input data by one place, or if we flip the data. In some cases, this is something we want in our machine learning algorithms, but more often than not, this behaviour is not a desirable outcome.

Given the downsides we have with regards to memory usage and non-spatiality in our multilayer perceptrons, we present Kunihiko Fukushima solution to solve both complications.

Convolutional neural networks are the most popular form for image recognition, segmentation, and classification

Convolutional networks work with filters and assign a weight to each pixel in the filters used. This use of filters significantly reduces the number of weights between layers, since we now have weights that are not dependent on the input size, and only dependent on the size and number of filters.

Given a $5x5$ filter

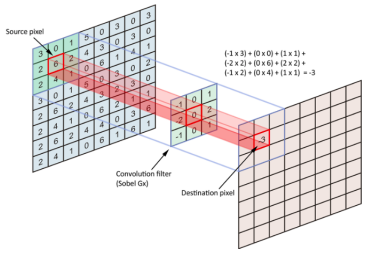

Figure 2.8: THIS IMAGE IS SHAME(LESS)LY taken from the internetz, draw own so the lawyers don't get you!

2.4.5 Additional layers

Activation layers

Pooling

Normalisation

Dropout

2.5 Neural Network models

2.5.1 Autoencoders

As we recall from earlier, an autoencoder is a type of neural network that tries to output a recreation of the output.

We can do this by having an encoder, $h = f(x)$, connected to a decoder, $r = g(h)$. An autoencoder has the job to set $g(f(x)) = x$ over the whole input, but in most cases this is not a practical program. We often gives the autoencoder the restriction that it has to map the input through a latent space that has a smaller dimension than the input dataset.
This is called an undercomplete autoencoder.

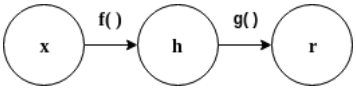

Figure 2.9: The general structure of an autoencoder, mapping **x** through **h** to an output **r**.

As with supervised classifiers we can use gradient decent to optimize the model. This is because we are trying to recreate the input **x** from out output $\widetilde{\mathbf{x}}$

This can simply be done by minimizeing the loss function

$$L(\mathbf{x}, g(f(\mathbf{x}))) \tag{2.10}$$

with for instance MSE with gradient decent.
Now we can transfer this to a more relevant example by making an image as input and use convolutions to reduce the dimensionality in the encoder and increase the dimentionality in the encoder.

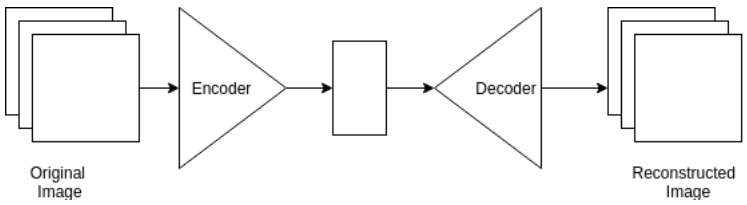

Figure 2.10: Convolutional autoencoder with an RGB image as input, and the reconstructed image as output.

2.5.2 Advaserial neural networks

This is explaining GANS, put me in the right place Now that we have looked at autoencoders we can take it a step further. generative advaserial models can be used as a generator of new data, and can have som reseblance to autoencoders 2.5.1, especially variational autoencoders

The difference lies in that advaserial networks is based on game theoretic scenarios in which a generator network is compeating agenst an advasery. The generator produces samples $x = g(z;\theta^{(g)})$, where g is the network given the weights θ. Then the discriminator network predicts if a sample is drawn from the dataset or from the generator. More spessific, it gives a probably given by $d(x;\theta^{(d)})$, and determins if x is from the generator or the data-set. Since we have two networks compeating agenst each other we can look at this as a Zero-sum game with the generators payoff is determined by $v(\theta^{(g)},\theta^{(d)})$, and the discriminators payoff is determined by $-v(\theta^{(g)},\theta^{(d)})$. *$v$ is here a function that is determined by both the sucsess rate of the discriminator and the generator, the most common used is*

$$v(\theta^{(g)},\theta^{(d)}) = \mathbb{E}_{x\sim p_{data}} \log d(x) + \mathbb{E}_{x\sim p_{model}} \log (1 - d(x)) \tag{2.11}$$

as derived from Goodfellow et al.

Lets look at a gan in detail.

UCNN?

2.5.3 Recurrent neural networks

LSTM

2.5.4 Contextencoders

Inpainting can also be done with advaserial models, and using a network trained to do the task of inpainting can be a lot more powerful than using

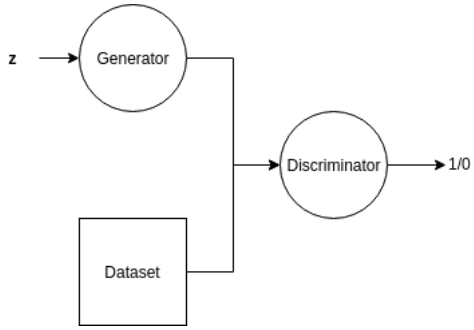

Figure 2.11: The idea behind a GAN. Here the generator saples from a random (Gaussian) distribution and generates samples that the discriminator classifies as real or fake

just an autoencoderor the naive methods. A contextencoder is building on the advaserial principle by using a generator/discriminator pair to fill in masked areas in an image.

The concept behind a Contextencoder is to take the whole image as input to an encoder/decoder pair and

2.5.5 CC-GANS

2.5.6 Pixel CNN

2.6 Explain how the ML-methods can be used with the polyps

When you work with machine learning a lot of the job is to make the data as clear as possible.
Imagine that you want to do something simple as reading an analogue clock. The s̈traight forwardẅay to do it is to make a convolutional neural network to look at the dials. This will require a much more complex network compared to if you could convert the angle of the dials to degrees and have that as an input to your model.

The trick is often to make the data as refined as possible. Further some of the techniques used is described.

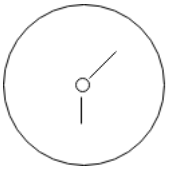

Figure 2.12: A clock needs a more complex network compared to just the degrees

2.7 The problem at hand

Now that we have the definition of machine learning and the current task, we can focus on the task at hand; finding polyps. In an ideal world[1] we have a Classification problem with only two classes: Non-polyp and polyp.

- SVM
- CNN
- random forests
- knn

[1] Ideal as in the only disease we could get in the GI tract was cancer originating from polyps which looked exactly the same

Chapter 3

Methodology

With our background in both machine learning and we can now look at how we want to solve the problems associated with setting up a system for medical diagnosis. We will first look at the language and packages used in the creation of this thesis. We will go in depth into the reasoning behind why we chose the tools and packages that became the foundation of the programs.

Then we will look at the setup of the complete program. Here we will go in-depth into both the different preprocessing algorithms, and take a look at the transfer learning network used during classifying.

3.1 Libraries

In this chapter, we will discuss the foundation of our code, important external libraries, and the setup and execution of our project. We will first discuss the programming language in question, give insight into the reasoning behind it. Then we will look into the framework used for machine learning, and in detail how it implemented in our programming language. Lastly, we will look into the wrapper we use to get a greater level of abstraction over our code, together with custom wrapper functions that are used by our wrapper.

3.1.1 python

When doing machine learning, the most popular languages, in no particular order, are: Python, Java, R, C++, and C . Some of these languages, like C and C++, are chosen for their speed, which is often a significant factor in Machine learning. Other languages, like R, is chosen because of its integration into the scientific community long before machine learning became a trend. The last group, consisting of Java and Python has gained popularity because of its

already big user base and user-friendliness. Python is also the winner when it comes to machine learning because of, like R, its integration into the scientific community. Right now Python is the leading language for machine learning. Driven by this, there is considerable focus into making it faster, to compete with already fast languages, like the C family.

Python is an interpreted, high-level, general-purpose programming language created in 1991. It, like many other modern languages, is object-oriented and supports functional programming.

Mainly because of the excellent support when it comes to machine learning, and the general ”easy to use and no compiling” we have chosen python as the base for our code in this thesis.

3.1.2 tensorflow

Arguably the biggest reason for the success of machine learning in python lies in Tensorflow.Tensorflow is a machine learning package developed by Google in and has since then become the leading framework for machine learning worldwide . Tensorflow is in use by companies like AMD, Nvidia, eBay and Snapchat.

Tensorflow is today a multi-language tool, but it had its origin in python. It is just in later years that other languages have gotten tensorflow support. The data flows through a graph network, where the objects in the graph describe the mathematical operations used in the machine learning, and the edges between graphs are the multidimensional arrays storing the weights associated with the operation in question. The name Tensorflow is a combination of the flow we experience during calculation and the tensors between the mathematical operations.

As stated, Python, and subsequently machine learning in Python, would be much slower than a counterpart in C. Because of this, Tensorflow works as a layer of abstraction to code running in the C language.

3.1.3 keras

One of the least attractive things with tensorflow is its unnecessary complexity. Even though Tensorflow offers more abstraction compared to running the code in pure C, the Tensorflow library can be unnecessarily complex. As a result of this, many external libraries try to simplify many of the complexities that accompany tensorflow. Libraries like TFlearn was made as a modular and transparent deep learning library on top of tensorflow. It gives a higher-level API to Tensoflow to reduce complexity and speed up experiments. The most

successful library for on top of Tensorflow is Keras . Just as TFlearn, Keras is a high-level package written in python. It is capable of running on top of TensorFlow, CNTK, or Theano, which is the tree most popular machine learning libraries at this time. From their website they state that their four goals when creating Kears were: **User friendliness.**
Modularity.
Easy extensibility.
Work with Python.

One of the core elements of Keras that makes it a better choice than just running, for instance, Tensorflow, is the concept of a model. A model in Keras is a way to organise the layers of the network in a more organised way, giving a better understanding of how the network is set up, and how each layer type contributes to the graph.

This thesis relies on Keras as a wrapper for tensorflow. As stated, the use of models and the simplicity of the language makes it an excellent choice of such a large project. Also, Keras has good support for convolutional operations which is the most used methods when managing images. Keras also has the most popular pretrained convolutional neural network models available in its package. *Since one of our primary goal is to see how well our datasets generalise to the real world, transfer-learning will be a great tool to forgo unnecessary training*

3.1.4 Custom functions for Keras and tensorflow

3.1.5 Additional packs in keras made by me

3.2 Describe code

3.3 Describe project

The whole project in

Chapter 4

Experiments

4.1 Datasets

To show and explain the experiments, we need to look at some datasets we can evaluate our results. In this thesis, we primarily use four datasets to train and evaluate our results.

4.1.1 kvasir

Automatic computerised detection of diseases has been a quintessential focus on the medical industry, but for a long time, a still fairly unexplored research area. As a response to this Simula Research laboratory together with , made the dataset Kvasir. It was originally made to improve medical practice and refine health care systems where similar dataset did not exist.

Kvasir is a dataset containing images from inside the gastrointestinal tract containing three anatomical landmarks, in the form of (A), (B) and (C). Also, the Kvasir dataset includes two categories of images related to endoscopic polyp removal, (D) and (E). And lastly, three classes, (G), (H), and (I), containing images without the anomalies, as a form of baseline. Medical professionals sorted the dataset, and it is made to be used for both single and multi-disease computer-aided detection.

At this time, there are two versions of the Kvasir dataset. V1 vs V2 The images are from both from the lower and upper GI tract.

4.1.2 Nerthus

4.1.3 CVC-356

4.1.4 CVC-12k

4.1.5 CVC-612

To discern the results of our experiments we introduce multiple metrics and tables to get an indication of our success. The main dataset we used for training, Kvasir, was split into k number of folds, using k-fold cross-validation. K-fold cross-validation is a tool used in statistics and machine learning to help to get an accurate representation of the data based on an average of the dataset. In machine learning, it is a powerful tool that can help with adapting to new datasets and prevent overfitting. Take the Kvasir dataset containing 8000 images with 1000 images from each class. We can choose a value for the number of folds, 'k', to be 6. This means that we split our dataset into six pieces before training. With the six pieces, we assign one of them as a test set, and we assign the four other for training and validation. We then train our data five times, using four folds for training and the last fold for validation during training. For each training run we rotate the validation set, so each of the five folds is used for validation once.

The advantage of this method is that we maximise the utility of the dataset. We find the distribution of the dataset that hopefully covers the most significant range of the unseen data.

4.1.6 presantation of data, confusion matrix

With k-fold cross-validation, we end up with the dataset that scored the highest during the final validation step. There are multiple ways to calculate metrics for how well a dataset is doing, but they all are a comparison of the predicted class versus the true class. Take for instance the case with the 8-class dataset Kvasir, where we predict an image to be of class 3, which in the real world be normal-z-line, while the actual True class is 5, and this might be esophagitis.

We can represent this as

```
[3, 5]
```

. Storing value pairs like this can very quickly get cluttered and unorganised. The most common way to store these value pairs is to use a confusion matrix. We initiate the matrix as a *NxN* matrix where N is the total number of classes.

```
[ 0  0  0  0  0  0  0  0]
[ 0  0  0  0  0  0  0  0]
[ 0  0  0  0  0  0  0  0]
[ 0  0  0  0  0  0  0  0]
[ 0  0  0  0  0  0  0  0]
[ 0  0  0  0  0  0  0  0]
[ 0  0  0  0  0  0  0  0]
[ 0  0  0  0  0  0  0  0]
```

After we have initisialised the confusion matrix, we add each value pair as to the matrix at is corresoponding postition. Given the pair

```
[3, 5]
```

we add one to the posistion corresoponding to (x=3,y=5). Another example could be given the pair where we guessed class 0 and the true class was 0 (

```
[0, 0]
```

), we add one to the matrix at position (x=0,y=0). Witch 2 examples we get the following confusion matrix.

```
[  1  0   0   0   0   0   0   0]
[  0  0   0   0   0   0   0   0]
[  0  0   0   0   0   0   0   0]
[  0  0   0   0   0   0   0   0]
[  0  0   0   0   0   0   0   0]
[  0  0   0   1   0   0   0   0]
[  0  0   0   0   0   0   0   0]
[  0  0   0   0   0   0   0   0]
```

As we fill in the matrix with more predictions we can start to draw conclusions from it. After appoximently 1600 evaluations, our result might at the end look like this.

```
[195  50   0   0   0   0   2   1]
[  4 148   1   0   0   0   0   0]
[  0   0 152   0   3  40   0   5]
[  0   1   0 198   0   0  13   4]
[  0   0   0   0 195   1   5   2]
[  0   0  47   0   1 159   0   0]
[  1   0   0   0   0   0 172   8]
[  0   1   0   2   1   0   8 180]
```

We can see that the highest numbers lies around the diagonal. This means that most of our results were classified correctly as values at the diagonal is the same x and y values, and subsequently a correct prediction. We can also discern something about the four primary metrics associated with a value in the matrix.

True Positive (TP): True positive for a specific class is when it is predicted positive, and the True label is also positive. In the Kvasir dataset, we have a True positive result if, for the class polyp, we predict a polyp.

True Negative (TN): True Negative is the opposite of true positive. Given the class Polyp from the Kvasir dataset, we guess that the image is not a polyp when the True label is non-polyp.

False Positive (FP): False positive is, given a True label we predict False. We often call this type of error a "Type 1 error". In the polyp case, this is the case where we predict a polyp given no polyp present.

False Negative (FN): False positive is the case where we fail to predict the class when it is True. This type of error is often called "Type 2 error". False Negative is, in our case, the least desirable outcome for our classes with pathological findings like Esophagitis, Polyps and Ulcerative Colitis.

For our multiclass confusion table, we can look at the four metrics like this

4.1.7 common Metrics

When evaluating our results, we use a set of common metrics used in the field of statistics and machine learning. The metrics are Recall (REC), Precision (PREC), Specificity (SPEC), Accuracy (ACC), Matthews correlation coefficient (MCC), and F1 score (F1).

Accuracy: Accuracy is one of the simplest metrics to understand. It describes how many of our predictions were correct out of the total predictions made. It is the most common metric given its simplicity both in calculation and understanding. In general when our data is balanced, and we only have a few classes, we can get away with using accuracy. In this project, we use accuracy during the training step as a metric of success.

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

Recall: Recall is the probability of detection . In a binary classification set, it measures the proportion of actual positives that are correctly identified as such. For our medical image classification, this metric can help us understand how our algorithms work by looking at classes where we have small anomalies in the images, like the case with polyps.

$$TPR = \frac{TP}{TP + FN} \tag{4.2}$$

Specificity: Specificity measures the proportions of our samples that were correctly identified as negative, when the true class were also negative.

$$TPR = \frac{TN}{TN + FP} \tag{4.3}$$

Precision:

F1 score:

Matthews correlation coefficient:

4.1.8 Singleclass vs Multiclass Metrics

The metrics presented are, in general, a solid way to present the validity of a model. However, not all metrics presented is the same when switching between single and multiclass classification. Metrics like Accuracy is designed to work for multiclass classification, given that there is only one way to calculate the score.

$$\frac{sum(diag(covariance_matrix))}{sum(covariance_matrix))} \tag{4.4}$$

The problem with multiclass metrics arises when there is more than one way to calculate the metrics needed, this can be for instance Recall and Specificity, where we have multiple ways to add together the classwise scores. The three most common ways to calculate the average are:

Micro average: calculates the mean value of each of the binary metrics and averages the result over the total number of samples. Micro average ignores all class frequencies and gives us a metric based on all samples gathered. Micro-averaging may be preferred in multilabel settings, including multiclass classification where a majority class is to be ignored.

Macro average: calculates the mean of each of the binary metrics, giving the same weight to each of the classes. Macro average gives importance to classes with few samples, and infrequent classes play the same roles as frequent ones. The disadvantage with Macro average is the fact that in the real world some classes often plays a more significant role than others, and doing especially bad on one of the classes can worsen the total result.

Weighted average calculates the mean of each of the binary metrics but gives a weighted sum for each of the scores before it is averaged. The weight of each class depends on the size of the true data samples. The weighted average gives us the advantage that small classes still count more than it would with for instance Micro average, but since it depends on the number of samples from each class it can end up more or less as a black box during calculation. Weighted average gives us the best of both worlds, but it lacks the intuitiveness from the two other classes.

With the three methods presented, we have chosen to Macro average our results. While both Macro and Weighted average would give a good indication given that not all our datasets are balanced, we argue that the weighted average would give metrics that are harder to explain when we are working on datasets with unbalanced classes.

In addition to looking at the Macro average of precision and recall, we want to look at specific cases of the classification. In many cases, we have multiple classes, where we are most interested in just one or a handful of the classes shown. For instance, a focus we have in this thesis is to give a score on how predictable polyp detection is, and on that case, we want to discuss the True positive rate (TPR) of the polyp detection and not the TPR of the non-polyp classes.

Take for instance the matrix shown in

```
[[10 1]
 [3 12]]
```

Here we can calculate the weighted average recall to be **x**. This can be an interesting observation in itself, but often the first or second True label is much more important relative to the other. In a more practical example: We are more interested in finding areas with polyps when we know they are present, compared knowing there is not a polyp in an area when none are present.

These Metrics becomes a more prominent topic when it comes to inpainting. With inpainting, we take areas with no relevant information and makes it into areas that are similar to the rest of the image. Given that we can inpaint over polyps by mistake, or that we might train our classifiers to not look in certain areas when classifying, we have an interest if also comparing single cases of recall and precision included to the average values.

4.2 Setup of experiments

We propose the following hypothesis.

```
When classifying images, we will get the best result when we have images with the l
Hence by removing areas with sparse information, we will see an increase in classi
```

and

```
When training a classifier, we will get a higher mode of generalisation of our resu
compared to not removing artefacts.
```

In this thesis, we will set up our experiments to test our two hypothesises.

4.3 format of experiments

4.3.1 Inpaint

4.3.2 Classifying

4.4 Inpainting Kvasir

4.4.1 Black corners

4.4.2 Green square

4.4.3 Text

4.4.4 Combination

4.4.5 Random masking

4.5 Kvasir - CVC612

4.6 Kvasir - CVC 12k

4.7 Kvasir - CVC356

4.8 Summary

Chapter 5

Result and Discussion

5.1 How to view the result?

5.1.1 The rate of success

What is a good result, how to measure?
FP,TN,FN,TP

Chapter 6

Conclusion

Chapter 7

Future Work