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

Acknowledgments

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

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. cite

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.

Statistics on cancer

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**

colorectal cancer

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.

polyps

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

preventative matters and early detection

-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

cite

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.

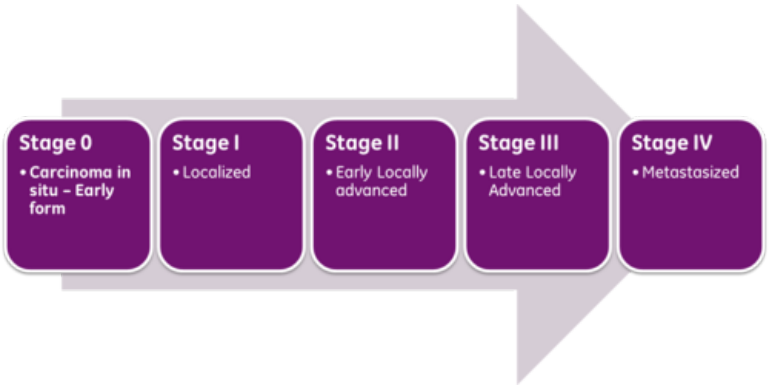


Figure 1.1: Stages of cancer from wikipedia

- 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

Simulas contribution to the pillcam project
Simulas EIR
* CAD ACD (computer-aided diagnosis, Automated computer diagnosis)

1.2 Problem definition

1.3 Research Method

1.4 Main contriburors and related work

1.5 Outline

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 the 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 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 Endoscopy and Colonoscopy

Gastrointestinal endoscopies are one of the most common medical examinations where the mucosa of the patient is visualised via a camera through the whole of the GI tract. Today, medical staff working with the visual screening of the intestinal tract use primarily two different methods: colonoscopy and gastroscopy. Colonoscopy is the practice of inserting a colonoscope into the rectum and moving through the large intestine towards the small intestine. Gastroscopy is the practice of inserting a camera via the mouth to get a visualisation through the stomach.

cite:https://www.ncbi.nlm.nih.gov/pubmed/24085631

The endoscopic tool used for this visualisation is made out of a flexible tube with a charged coupled device (CCD) working as a camera at the end. In addition to the light sensing chip, there is also an optical fibre to transport light to the camera. At the other end, the colonoscope is connected to a device that records the video, and a light source for the optical fibre. The video from the CCD is shown live for the medical staff for the doctors to analyse.

2.1.2 pill cam

2.1.3 Medical images/data/other

2.1.4 Systems in place for detection

2.1.5 summary

We have now looked at the medical perspective of colonoscopy. We see that the practice has not changed since the introduction of the colonoscope and that the best chance of detection is manually looking at images by professional medical staff. We have also taken a look into the automatic detection methods being experimented with today.

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 Mirmir works, we need to look at how Machine learning helps with the detection of the anomalies the medical staff are after.

cite

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.

fix

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

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

more

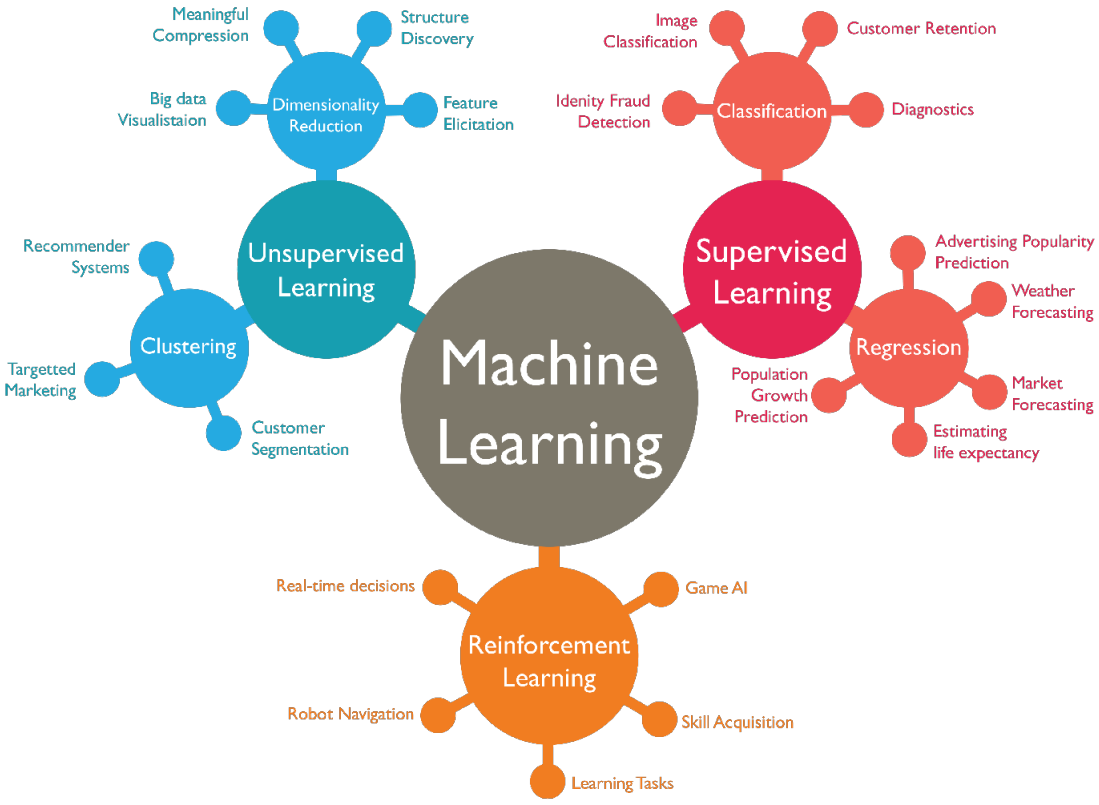


Figure 2.1: The three main types of machine learning and their subtypes

supervised machine learning

finish and cite

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.

Stuart J. Russell, Peter Norvig (2010) Artificial Intelligence: A Modern Approach, Third Edition, Prentice Hall ISBN 9780136042594.

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

steps

more info

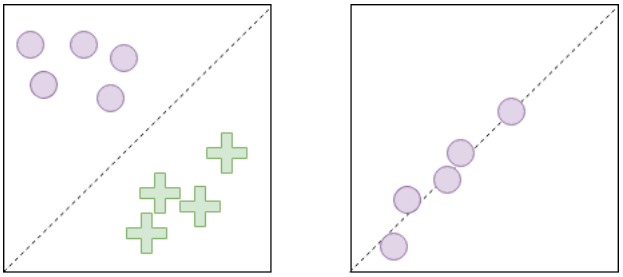


Figure 2.2: Left: Example of binary classification. Right: Example of regression

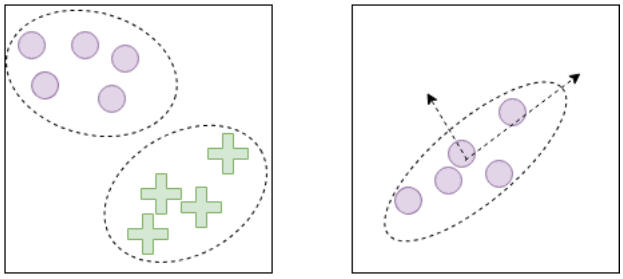


Figure 2.3: Left: Example of binary clustering. Right: Example of principal component analysis

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 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

more

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

fast

no training

talk about it as a clustering method, classification

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.

proposed by

cite

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.

find out how much I need to write here

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.

cite

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.

perhaps NN

might be own chapter

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

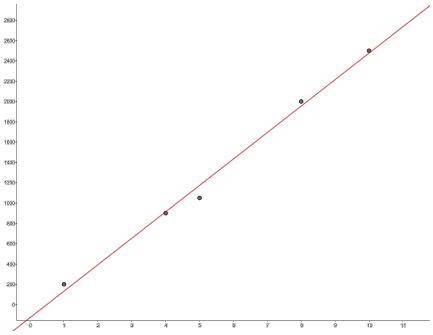


Figure 2.4: Example of linear regression in Geogebra Here the red line is the best approxmiation of a y value, given an x value.

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

tell that this is loosely based on deeplearningbook, basic machine learning

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.

cite geogebra

. 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 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.

more about GDEC

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):

cite Wikipedia GD

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

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.

who

find when

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

non-linear problems

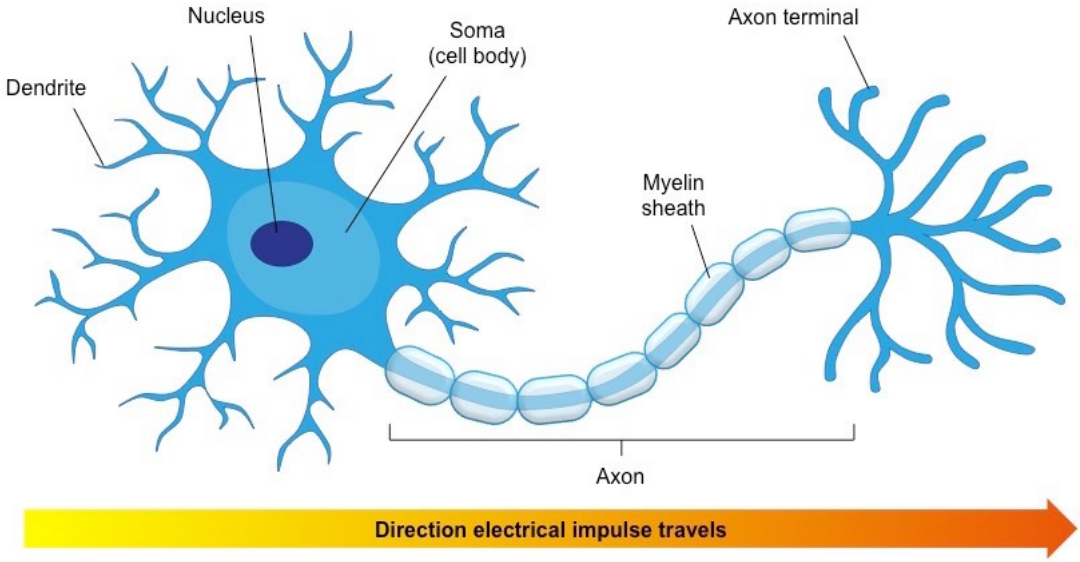


Figure 2.5: https://socratic.org/questions/how-is-a-neuron-adapted-to-perform-its-function

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.

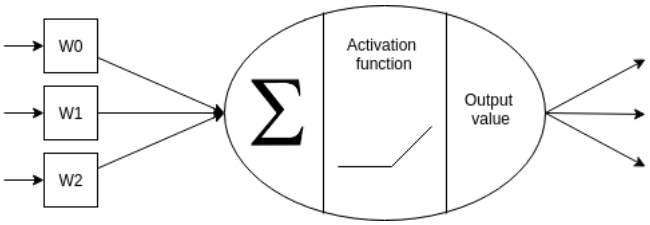


Figure 2.6: Simple perceptron

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)

cite

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

talk about why we use an activation function

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.

cite

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

talk about the single neuron made by that dude

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

find formula

2.4.3 Feed forward and backpropagation through a neural network

looked at how MLP works, let us look at back-and-forward prop

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

ref chapter on MLP

insert math

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

fix this

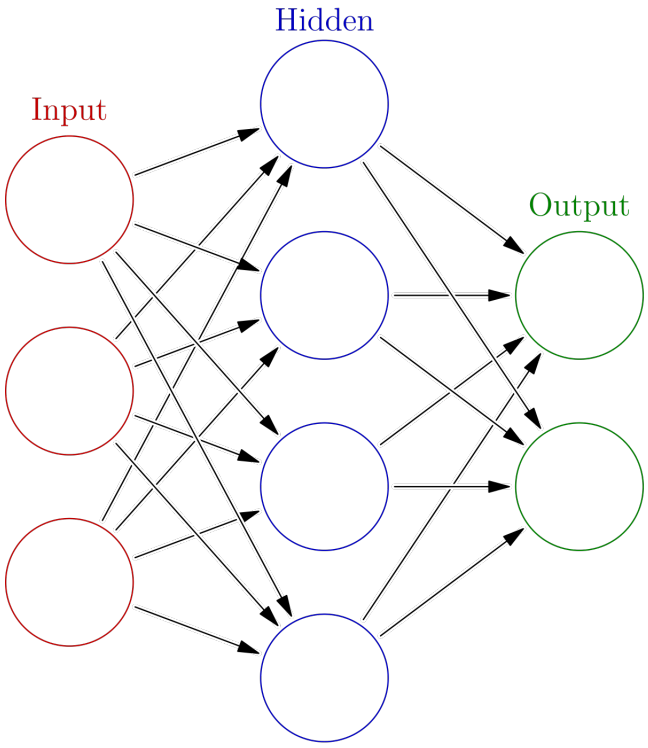


Figure 2.7: THIS IMAGE IS SHAME(LESS)LY taken from the internetz, draw own so the lawyers don't get you!

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.

cite

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

cite, IEEE paper got cites

When building a convolutional neural network we often use multiple layers stacked on opt of each other to give the network traits that a regular multilayer perceptron could not achieve. By far the most essential layer of the convolutional neural network is the convolutional layer.

Convolutional layers

Convolutional networks work with filters as opposed to perceptrons with weights assigned before and after the input in the vertices between perceptron. Convolutional layers assign a weight to each position in a special filter matrix.

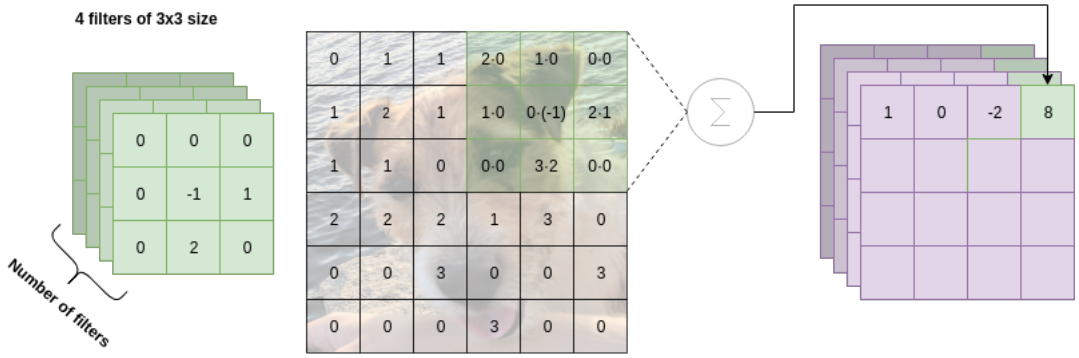


Figure 2.8: The values calculated when a convolutional filter after 4 sliding window operations, here the number of inputs does not represent how many inputs there usually is in an image

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 this matrix size.

The three main parameters of a convolutional layer are the number of filters, kernel size and strides. When making a convolutional layer, we start by pseudo-randomly initialising a $(F \times K \times K)$ matrix as our weights. In this matrix, F is the number of filters in the convolutional layer and K is the filter size. We can think of this as F filters of size $(K \times K)$ stacked on top of each other. When applying the convolutional layer to an input vector, we take each of the F filters and slide it over the image. For each position of the filter, we multiply the image value with the filter and sum the result. The scalar made by this multiplication is the value passed on to the next layer at that specific point. Figure 2.8 shows the this convolution process after 3 sliding operations with a (3×3) filter.

As we can see from this architecture, we can only change when weights in the filter matrix, as there are no other variables in the convolutional operation. Using this sliding window technique gives us the benefit that the filter only gathers information from the local area, and subsequently makes the convolutional operation non-spatially dependent.

Activation layers

As we discussed in chapter 2.4.1, in addition to summing the inputs and passing them on, we need to apply an activation function to the output. The same problem with our desire to make non-linear problems apply in the CNN model. To apply an activation layer to a CNN, we take every value in the matrix and apply the activation function separately to every data-point.

Pooling

Pooling layers, often also called downsampling layers, are commonly also found in CNN's. Their primary role in the network is to reduce the spatial size of the data provided, and thus reducing the number of weights and variables used by the network. The two most common pooling operations are max pooling and average pooling. Both methods resemble convolutions in the way that they apply their function to a sliding filter over the data.

cite

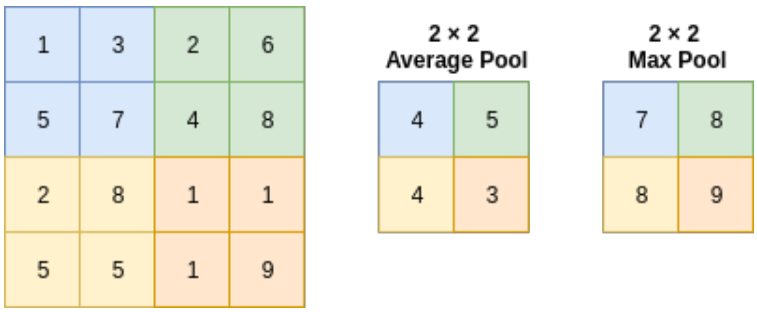


Figure 2.9: Both max and average pooling done on a 4×4 matrix

Just like in Figure 2.9 the most common pooling parameters are of filter size 2 and stride 2. Using this configuration leaves every point of the input checked once, and the resulting output size is halved. Pooling layers do generally not have any weights associated with them. This lack of weights means that the pooling layer routes the signal back without changing it during backpropagation.

unpooling

Normalisation

Normalisation layers are one of the newer addition to the CNN models. The normalisation layers, as with the pooling layers does not contain any weights, ad subsequently only change the data based on the statistics of the data.

In CNN models the most used type of normalisation is batch normalisation.

- Learning faster
- Increase Accuracy

Dropout

A major problem in machine learning is the act of overfitting the data. We say that a model is overfitting to the data when it learns the bias that comes with the dataset.

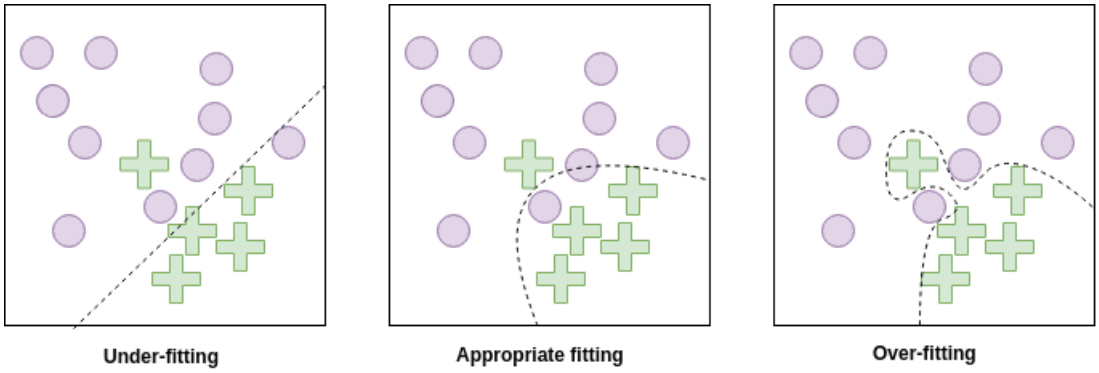


Figure 2.10: Three cases of data boundary prediction. In most cases we desire appropriate amount of fitting to our dataset to keep generalisation

Dropout, first proposed by Srivastava et al., in 2014 was used to prevent overfitting of neural networks (As seen in figure 2.10). Dropout layers in the CNN will randomly cut a certain amount of connections in a layer. The amount is usually between 25% and 50% of the connections. The result of the use of dropout is that the network can not rely on only a few numbers of weights during training since there is a chance the input from the weight will be cut at random intervals.

This random cutting forces the network to spread out the essential weights throughout the network. Even though dropout does not explicitly change the weights, it implicitly forces the stong weights to be eavenly distributed throughout the network. Dropout does, in practice, the same as an L1 or L2 regularizer.

2.5 Complex Neural Network models

At this point, we have looked at the most crucial parts of a convolutional neural network individually.

2.5.1 Autoencoders

In the realm of image-generating one of the simplest networks to train and use is the Autoencoder. In 19999 Mathias et al. proposed a encoder-decoder network where the task was to recreate the original input as an output.

In this thesis we are using an autoencoder to do this thing

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 give 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 type of network is called an under-complete autoencoder.

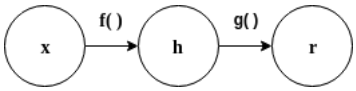


Figure 2.11: The general structure of an autoencoder, mapping **x** through **h** to an output **r**.

As with supervised classifiers, we can use gradient descent to optimise 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 minimizing the loss function

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

explain MSE, and general loss on an earlier time

with for instance MSE with gradient descent.
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 dimensionality in the encoder.

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 adversarial models can be used as a generator of new data and can have some resemblance to autoencoders 2.5.1, especially variational autoencoders

The difference lies in that adversarial networks is based on game theoretic scenarios in which a generator network is competing against 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 competing against 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 success rate of the discriminator and the generator, the most*

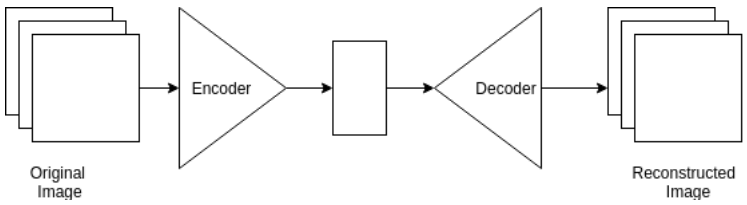


Figure 2.12: Convolutional autoencoder with an RGB image as input, and the reconstructed image as output.

commonly 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.

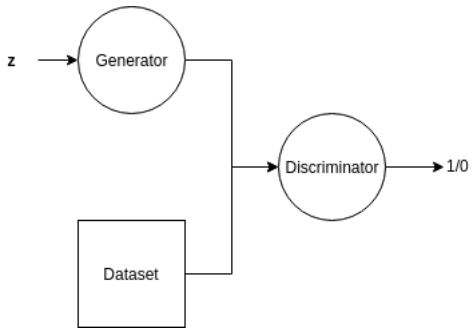


Figure 2.13: The idea behind a GAN. Here the generator samples from a random (Gaussian) distribution and generates samples that the discriminator classifies as real or fake

Contextencoders

Inpainting can also be done with adversarial models, and using a network trained to do the task of inpainting can be a lot more powerful than using just an autoencoder or the naive methods. A contextencoder is building on the adversarial 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

ref

ref

finish

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 as 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.

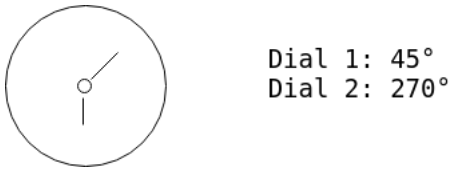


Figure 2.14: A clock needs a more complex network compared to just the degrees

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

Here I need to talk about how preprocessing might help the problem, and make this a springboard into the next section: the problem at hand

adress the problem with lack of data, and talk about how machine learning can help, machine learning.

2.7 The problem at hand / Summary

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.

medical background

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 higher 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

cite

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

cite

year

cote

something about projects with python, and how may uses

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.

cpu vs gpu

, CNTK, or Theano

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

cite TFLEARN

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.

cite keras

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.

more

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, tensorflow and python

CWFC layer

Spectral conv?

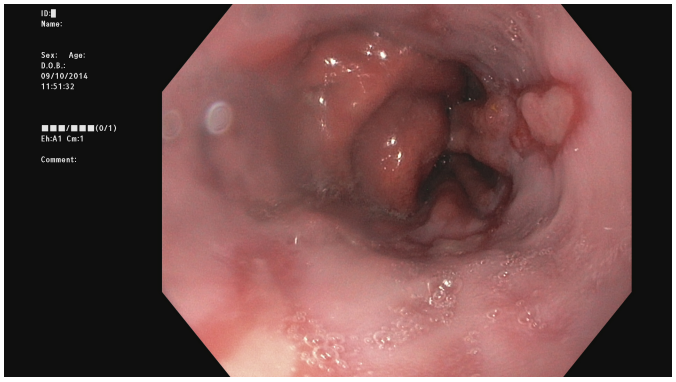
Masklaod

self attention

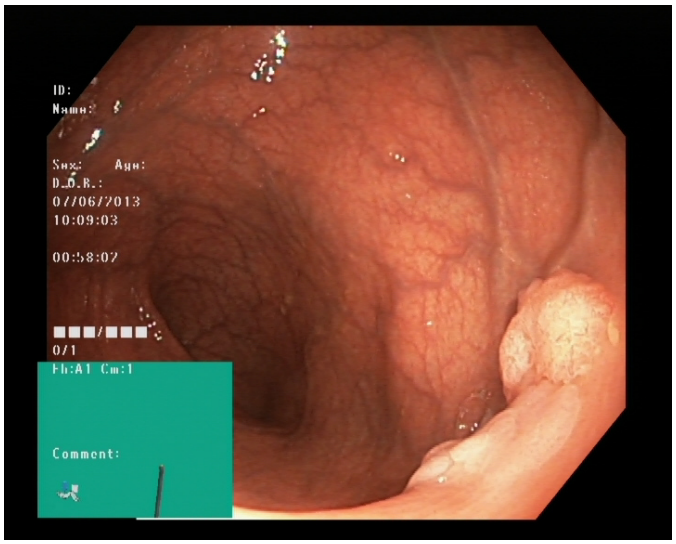
masked loss

subpixel

3.1.5 Additional packs in Keras made by me



(a) $y = x$



(b) $y = 3sinx$

Figure 3.1: Three simple graphs

3.2 Design of the inpainting experiments

When inpainting we have multiple hypotheses we want to test to see how it affects the results. We first want to set up a platform where every dataset is made with the same parameters, except for the dataset-specific parameters that define the dataset. Based on the hardware limitations

more

When generating the datasets, we use the Kvasir dataset as a base exclusively. When using only the Kvasir dataset, we have the CVC sets for testing. This selection of image source was made intentionally to have a fundamentally different test and training set, and the more differences between testing and training set the more of an indication of generalisation.

Figure shows three different image types from the Kvasir dataset. Image (a) shows the image class esophagitis. This image shows one of the main problems with the Kvasir dataset when it comes to dataset specific artefacts. In addition to the cut corners, we have an extra wide area to the left of the image containing non-relevant information like name, sex, and other comments. We can recall from chapter that the area to the left does not contain any relevant information for the classification and subsequently can only give false information during classification. Another example of dataset-specific artefacts is in the image (b). In this image, we have the same cut corners as in most colonoscopy images. However, we also have an additional green square in the bottom left corner.

ref

find where

Compared to the CVC images we have images in the Kvasir set with this square and with additional text, both outside the image, and on top of the image.

We propose different types of inpainting to prove or disprove our two hypothesises.

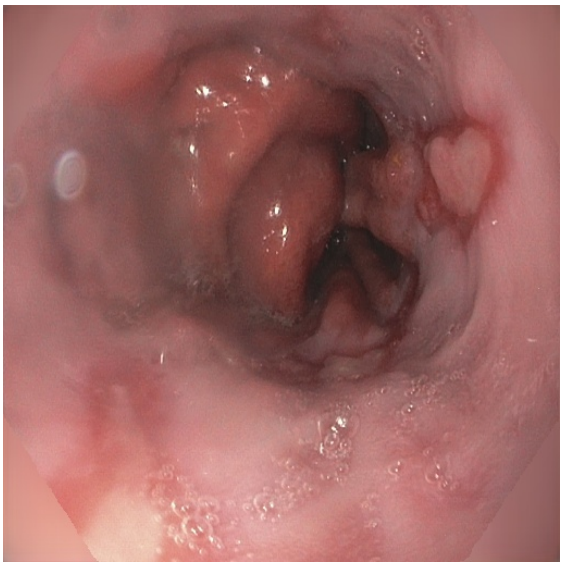
3.2.1 Removing black corners

The most straightforward experiment to conduct is to test how the removal of the black corners will affect the result. The black edges around the images in the Kvasir set is also present in the CVC set, and the Nerthus set too. By removing the black corners around the image, we do not change Kvasir specific artefacts, but according to our first hypothesis, we believe we will get a higher classification accuracy since this removes areas with sparse information.

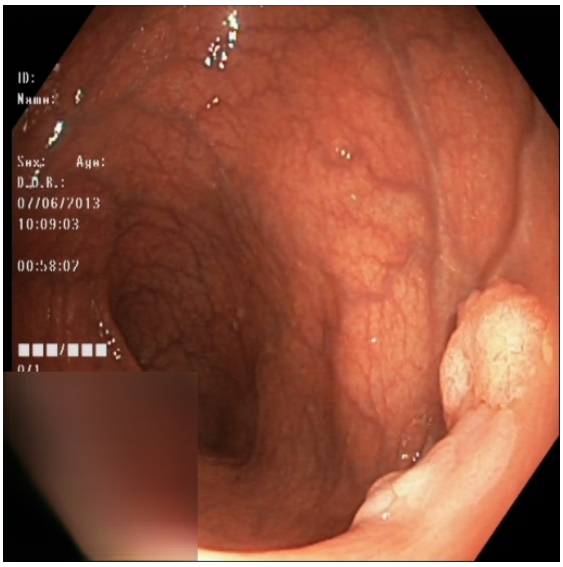
When classifying images during the testing stage, we need to take this mask in to account regarding the removal of black corners in the test set.

1. We can either do the same masking and inpainting on the test set.
2. We can crop the image in a way that removes the black corners without inpainting.
3. We can forgo modifying the test data and just run the test set as is.

Method 1) was tested in the paper by Kirkerød et al. in the Mediaeval 2018 conference, where both the training and test data were augmented. From the experiments run at the mediaeval conference, we have some essential information we can take into account. Training data used to train the preprocessing tools were data from the same dataset as Kvasir, giving the project and thesis an overlap in the data distribution during training. The evaluation of the test set was done in beforehand, and not live. Also, there was no focus on evaluation speed when running the classification on the test set.



(a) $y = x$



(b) $y = 3sinx$

Figure 3.2: Three simple graphs

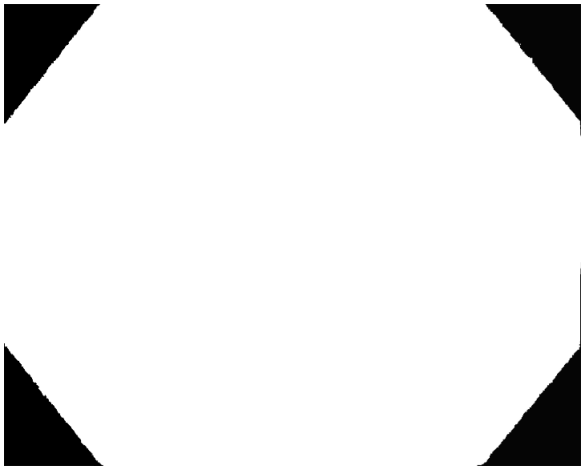


Figure 3.3: The automatially generated mask used to remove the black corners

The Medico test set used in the Mediaeval conference was also gathered from the same distribution as the Medico test set. This use of the same distribution means that if we are going to compare methods of masking the test set, we should only look at the Kvasir dataset when comparing the masking methods, given the CVC dataset will not give sufficient comparability. Also, given that our goal with the medical classification is, in the future, to have the opportunity for live classification. We have chosen in this thesis not to convert medical images during testing given the time used during a live evaluation would not be suitable.

more about the result of testing inpainting.

Method number 2) was the proposal of cropping the images during evaluation. Thambawita et al. did similar methods in the Mediaeval 2018 conference, but had, in addition, the same cropping during training. In the paper we can see that this method worked with great success. The operation of just cropping images is also multiple times as fast as inpainting images, so it is feasible for a live recording.

find Vajiras paper

The last proposed method is not to augment the images during testing. This method, since we are not preprocessing the test data at all, the fastest when it comes to live classification. Without the augmentation, we risk getting a lower classification score.

In this thesis we will use method number 3). The decision is based on the fact that the final product would most likely be used in cooperation with medical staff, and hence the image should be as close to the feed from the pillcam/colonoscope. Method 2) might also work, but we remove data from the image that the medical staff might need for their classification.

3.2.2 Removing green squares

The next major area in question is the removal of the green squares. The experiment would see if the removal of the green ḿinimapẃould affect the classification score.

Figure 3.4: The automatially generated mask used to remove the black corners

The removal of the green squares hypothesis 2

3.2.3 Removing black corners

3.2.4 Removing black corners

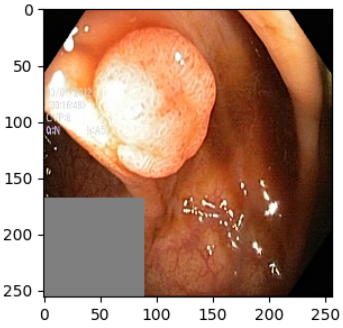
3.3 Design of the transfer learning experiments

3.4 Describe code

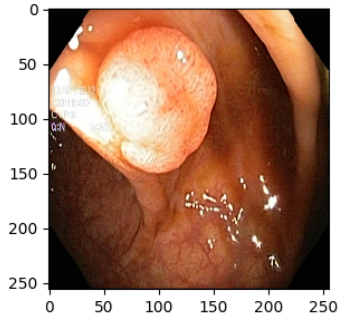
3.4.1 autoencoder

x et al

The autoencoder we used in this thesis bears some resemblance to the standard autoencoder by . To describe the model we will look at the example where we try to inpaint the green square in the image, and nothing else. To train the autoencoder for inpainting, we divide the dataset in two, first images with the green square and images without the green square. We discard the images with the green square since they are not viable for training. The resulting dataset will only contain images without green sources.



(a) Image the autoencoder receives as an input



(b) The missing part the autoencoder tries to replicate

Figure 3.5: A standard image taken in by the autoencoder

The next step before training is to cut the images according to the mask provided. Figure 3.5a shows what the finished masking looks like, and 3.5b shows what we want to achieve after training.

We feed 3.5b into the autoencoder consisting of convolutional layers, leakyReLu layers, and a tanh layer.

more

remember to talk about loss

3.4.2 Generative adversarial network

The gan used the same generator discriminator elements as the Goodfellow gan, but does not take in noise

the best version does

3.4.3 Transfer learning classifier

The dataset we are making with our generators needs to be classified.

The classifier we use is based on the idea of reusing networks we already know apply well to the real world. We have made a classifier that, by default, use one of the pretrained networks provided by the Keras framework.

Table 3.1 shows the pretrained networks available to load in the Keras framework. When training we did some extra steps at the end, namely added global average pooling and a fully connected layer with the desired number of outputs (usually eight classes, and eight outputs)

3.5 Describe project

Here i am going to explain the projects

Table 3.1: Datasets provided by keras

Model	Size	Top-1 Accuracy	Top-5 Acc	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
ResNeXt50	96 MB	0.777	0.938	25,097,128	-
ResNeXt101	170 MB	0.787	0.943	44,315,560	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

3.6 Summary

At this point, we have described the reasoning behind using python, tensorflow and Keras for our machine learning. After this we looked into the filters used for training our models, and what type of preprocessing we wanted to do in addition to the inpainting. We looked at the advantages and disadvantages of preprocessing the Trest data, as opposed to just preprocessing the training data.

We then went more in-depth into how the three main programs were built up, and how they differ from their original sources. First, we looked at the preprocessing methods, AE and GAN. We looked at how they learned from the samples, and we went more in-depth talking about the crucial layers for each model.

At last, we ended up giving a summary of the project as a whole, following a the Kvasir dataset from its original source into the generation of the six new datasets, and how each one of them was classified with all the different

parameters to show the dataset-specific rate of success.

Chapter 4

Experiments

In the previous chapter, we described our methodology. We presented the framework we used for our setup, and we went into detail on the configuration of the experiments, and the project as a whole describing both the masking and the neural networks. In this chapter, we will start with the description of the datasets used on the results and the metrics chosen as a reference point for the evaluation of our results. Finally, for each model, we will evaluate it for the datasets we have presented.

4.1 Datasets

To show and explain the experiments, we need to look at the datasets we have to evaluate our results.

In this thesis, we primarily use four datasets to train and evaluate our results.

something about Simula already researching on this

4.1.1 kvasir

As we recall from the introduction chapter, oesophagal, stomach and colorectal cancer accounts for about 2.8 million new cases and 1.8 million deaths per year, and this number is increasing as the population gets older. With automatic detection of diseases by use of computers being prominent, but a still unexplored field of research the dataset Kvasir was made. Kvasir is a Multi-Class Image Dataset for Computer Aided Gastrointestinal Disease Detection made in Norway. The data is gathered from the Vestre Viken Health Trust, and it contains not only polyps but also two other findings, two classes related to polyp removal and three anatomical landmarks in the GI tract.

The data is collected using endoscopic equipment at Vestre Viken Health Trust in Norway. The Vestre Viken Health Trust is a collection of 4 hospitals, and it provides health care to 470.000 people. One of the more prominent hospitals in this group is Bærum Hospital. It has a large gastroenterology department where the data originates, and more data will be provided in the future. The data from Bærum Hospital has been annotated by medical experts and the Cancer Registry of Norway before its inclusion in the Kvasir dataset, making the data thoroughly marked and checked. The Cancer Registry of Norway work provides new knowledge about cancer prevention and cancer treatment. It is part of the South-Eastern Norway Regional Health Authority and organised as an independent institution under Oslo University Hospital Trust. The Cancer Registry of Norway is responsible for the national cancer screening programmes with the goal to ultimately prevent cancer death by discovering cancers or pre-cancerous lesions as early as possible.

Kvasir is a dataset containing images from inside the gastrointestinal tract containing three anatomical landmarks, in the form of the Z-line (a), the Pylorus (b) and the Cecum (c). Also, the Kvasir dataset includes two categories of images related to endoscopic polyp removal, Dyed and Lifted Polyps (d) and Dyed Resection Margins (e), and lastly, three classes, Esophagitis (g), Polyps (h), and Ulcerative Colitis (i), containing images of pathological findings.

Anatomical Landmarks

The definition of an anatomical landmark is a feature that is easily recognisable through an endoscope. For the medical staff they are essential for navigating the GI tract, and the assist as a reference point both for a physical location, and to describe the position for a relevant finding. It is often relevant for a complete endoscopic report to contain image documentation and a description of these three landmarks.

Z-line The Z-line is the sphincter between the oesophagus and the stomach. This sphincter is an endoscopically recognisable border where the white mucosa meets the red gastric mucosa from the stomach. When performing endoscopy an assessment of the Z-line is important to determine whether disease is present or not. A malfunctioning sphincter can, for instance, lead to diseases like gastroesophageal reflux. . Figure **??** shows a normal z-line taken from an endoscopy.

cite

Pylorus We call the barrier between the stomach and the upper part of the small bowel for the Pylorus. This sphincter is responsible for letting food from the stomach into the GI tract, and to keep the stomach acid from flowing into the

bowels. Medical staff inspect both sides of the pylorus when doing a complete colonoscopy and endoscopy. This inspection is crucial to identify findings like ulcerations, erosions or stenosis. Figure ?? shows an endoscopic image of a healthy pylorus taken from inside the stomach.

cite

Cecum The last anatomical landmark used in gastronomy is the cecum. It is an intraperitoneal pouch that is considered to be the beginning of the large intestine. This area in the GI tract is considered the final stop for a standard colonoscopy as it is the last quality indicator for the procedure. Being the last landmark, the identification and inspection of this area are one of the most crucial steps in the whole inspection. Figure ?? shows a normal cecum.

4.1.2 Pathological Findings

Pathological findings are areas or objects with an abnormal or unwanted feature. When looking at it from the oral entrance, pathological findings takes the form of inflammation, while in the lower GI tract it takes the form of, for instance, polyps. Pathological findings are the group of images that are especially important to detect and classify as they impose the most significant risk for the patients well being.

do not like the ending

Esophagitis Esophagitis is an inflammation oesophagus near the Z-line. Usually, the oesophagus is composed of a mucosal lining and circular smooth muscle fibres, but when a patient has a condition like a gastroesophageal reflux disease the area can become inflamed. Clinically, detection of this inflammation is necessary for treatment initiation to relieve the symptoms and to prevent further development that can lead to more complications.

polyps Polyps are lesions in the bowel detectable as mucosal outgrowths. There are multiple types of polyps that are either flat, elevated, or pedunculated. Polyps are distinguishable from normal mucosa as they have a different colour and surface texture. Most of these bowel anomalies are harmless, but they have, as discussed in the introduction, the possibility to become cancerous. There is a big focus dedicated to the removal and detection of there polyps as they are a significant threat to cancer of not treated in time. Because of the different shape and rate of outgrowth, medical staff overlook a small, but significant part of the polyps during routine procedures. This rate of error was the most significant driving force for investing in computer-aided medical diagnosis, and subsequently creating the Kvasir dataset.

Ulcerative Colitis Ulcerative Colitis is the last pathological finding type in the Kvasir dataset. It is a chronic inflammatory disease affecting the large bowel. Ulcerative Colitis may both come from psychological and physiological factors, but the origin of the disease is often unknown. The degree of inflammation is a factor when performing medical diagnosis, though in the Kvasir dataset we only have one class of ulcerative colitis. A severe example of ulcerative colitis can is seen in figure **??**. Here we have an evident inflammation, and thus this would most likely be classified as severe.

4.1.3 Polyp Removal

cite

The last groups are concerned with the removal of polyps. A technique often used in polyp removal is endoscopic mucosal resection (EMR). This procedure includes the injection of a liquid underneath the polyp, lifting it from the underlying tissue. After the injection, the medical staff use a snare to wrap around remove the polyp. Using this method of lifting and snaring, the medical staff reduces the risk of damaging the surrounding tissue by a large margin.

Dyed and Lifted Polyps The first of the two classes concerning polyp removal is the Dyed and Lifted Polyps class. This case is the polyp after it has been lifted from the surrounding tissue, and the next step is to use a snare to remove it. Figure **??** shows a polyp after the injection liquid is applied.

Dyed Resection Margins The resection margin is important to evaluate after the removal of a polyp, as we want to be sure the entirety of the are is clean. If the procedure did not eradicate the polyp, the residual polyp tissue might lead to continued growth, and in the worst case malignancy development.

do we do Nerthus?

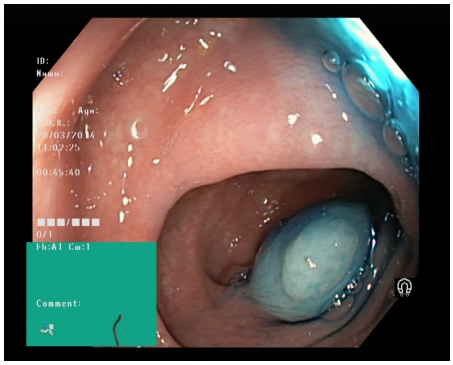
4.1.4 CVC-356

The

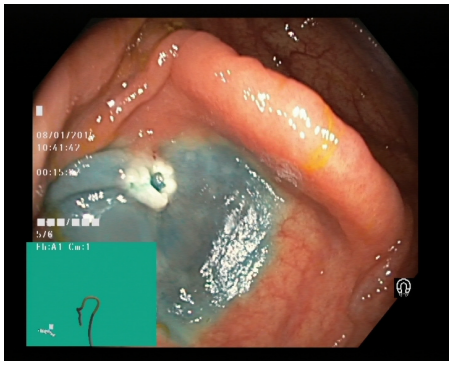
4.1.5 CVC-12k

4.1.6 CVC-612

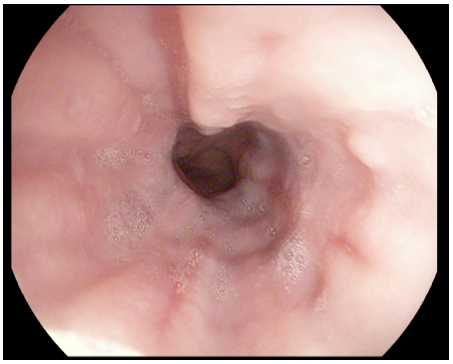
esdtgfgv



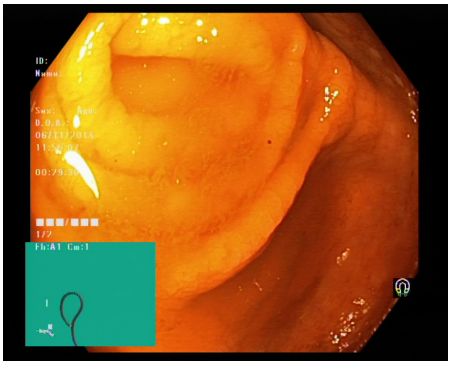
(a)



(b)

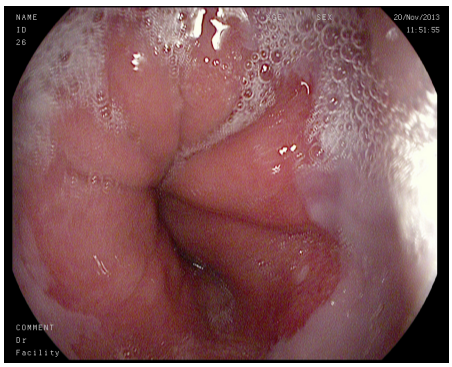


(c)

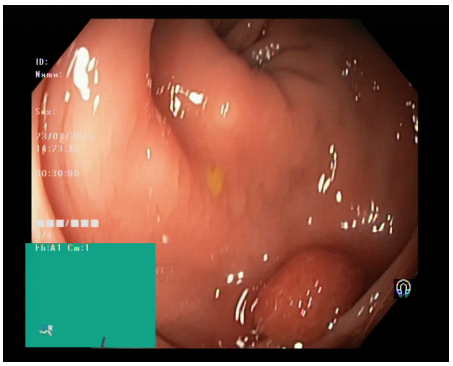


(d)

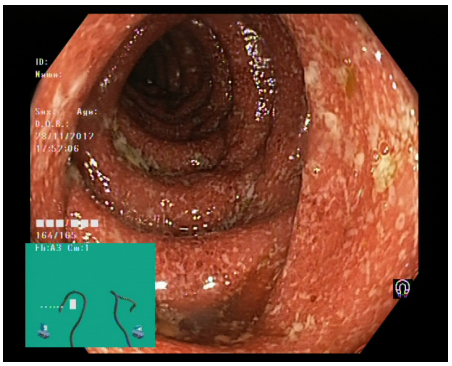
(e)



(f)



(g)



(h)

Figure 4.1: New text here

4.2 Metrics

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.

img of k fold

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.

4.2.1 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.

rewrite

We can represent this as

$$\begin{bmatrix} 3 & 5 \end{bmatrix}$$

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.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

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

$$\begin{bmatrix} 3 & 5 \end{bmatrix}$$

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

$$\begin{bmatrix} 0 & 0 \end{bmatrix}$$

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

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

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.

$$\begin{bmatrix} 195 & 50 & 0 & 0 & 0 & 0 & 2 & 1 \\ 4 & 148 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 152 & 0 & 3 & 40 & 0 & 5 \\ 0 & 1 & 0 & 198 & 0 & 0 & 13 & 4 \\ 0 & 0 & 0 & 0 & 195 & 1 & 5 & 2 \\ 0 & 0 & 47 & 0 & 1 & 159 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 172 & 8 \\ 0 & 1 & 0 & 2 & 1 & 0 & 8 & 180 \end{bmatrix}$$

We can see that the highest numbers lie around the diagonal. This centralisation means that most of our results were classified correctly, as values at the diagonal are 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

4.2.2 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

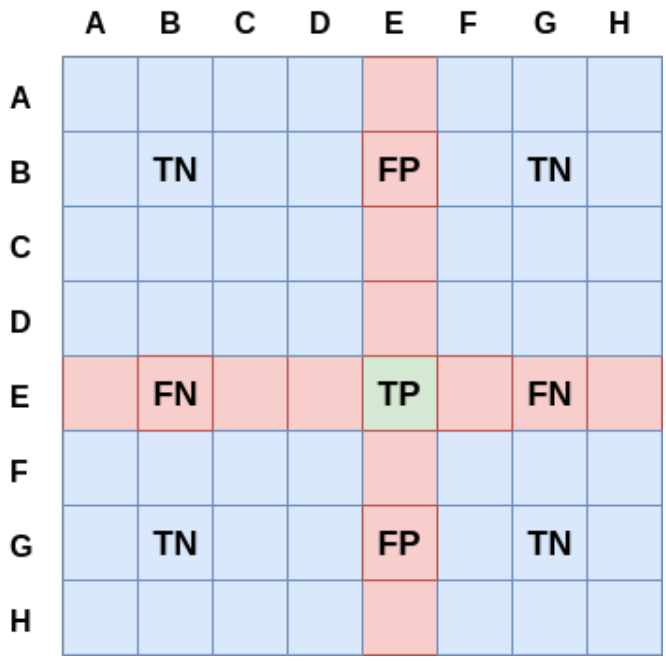


Figure 4.2: Confusion matrix with eight classes, here True positive is marked in green, False Negative and False positive marked in red, and False negative in blue.

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: F1 score

Matthews correlation coefficient:

4.2.3 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 fig

```
[[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.

4.3 Setup of experiments and hardware

We propose the following hypothesis.

When classifying images, we will get the best result when we have images with the least amount of sparse information. Hence by removing areas with sparse information, we will see an increase in classification performance compared to not removing areas.

and

When training a classifier, we will get a higher mode of generalisation of our results when removing the dataset specific artefacts compared to not removing artefacts.

In this thesis, we will set up our experiments to test our two hypothesises. We divide our work into two parts, inpainting and classifying. First, we will look at the process of inpainting in detail, and inspect the results we have. We will look at how parameters affect the results, and how different networks will differ in the generating process. more

After a rundown of the generation of the custom dataset trough inpainting, we will show how the classification scores. Here we look at the datasets generated by inpainting and compare them with a base case.

find MCC chapter

find dataset chapter

2x recall

Our primary goal in this thesis is to see if any of the generated datasets can help with classification. We will both compare the different areas inpainted, and the method used to inpaint the images. In addition to just comparing accuracy, we will also look especially at Recall of the polyp class, and the MCC score. As mentioned in chapter and the , we have datasets which are unbalanced, and subsequently, need a more representative score compared to F1 or accuracy. We can also recall that the recall score gives us a probability of detection in a binary class situation. We will use this to see if any one of the generated datasets modifies this score in any manner.

We want to end this chapter with a small experiment to see if network types affect the recall of polyps if convolutions are used compared to dense connections. We will use this to conclude if it is the network type of the inpainting that has the best effect when it comes to a good recall score for polyps.

hardware

The Hardware used for the generation and calssification is as follows.

Table 4.1: Hardware for things

Part	Type
GPU	GTX1080TI
CPU	I3?
RAM	16 GB

4.4 Results of the Inpainting

We will first look at the inpainting data. Here we have 3 masks used, and 2 different networks gnererating the images. This gives us the in total six augmented datasets.

4.4.1 Black corners

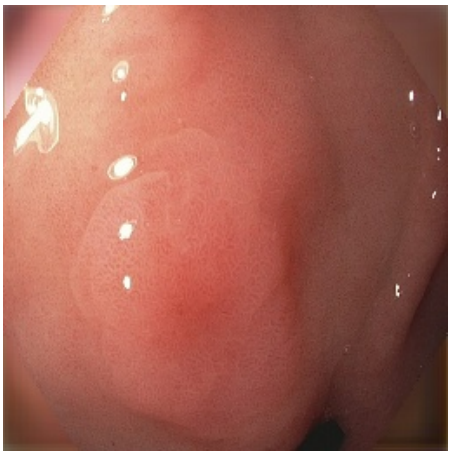
4.4.2 Green square

4.4.3 Combination

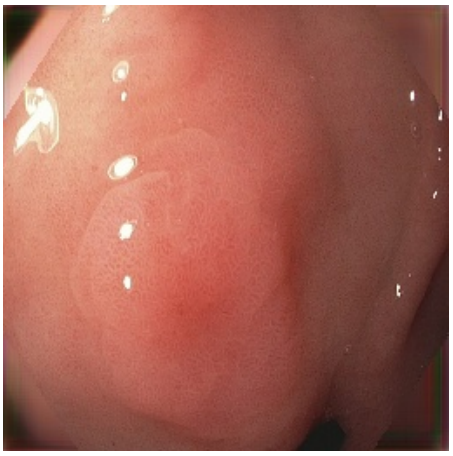
Table 4.2: Details of all datasets used in the experiments.

BC: Black corner. **GS**: Green square. **BC+GS**: Black corner and Green square

Dataset labels	Size	Inpainted area	Generator network used
I - Base Case	256x256 px	-	-
II - Autoencoder with black corner	256x256 px	BC	Autoencoder
III - Autoencoder with green square	256x256 px	GS	Autoencoder
IV - Autoencoder with both	256x256 px	BC+GS	Autoencoder
V - GAN with black corner	256x256 px	BC	GAN
VI - GAN with green square	256x256 px	GS	GAN
VII - GAN with both	256x256 px	BC+GS	GAN



(a) Image from the polyp class at 256x256 px, generated by an autoencoder



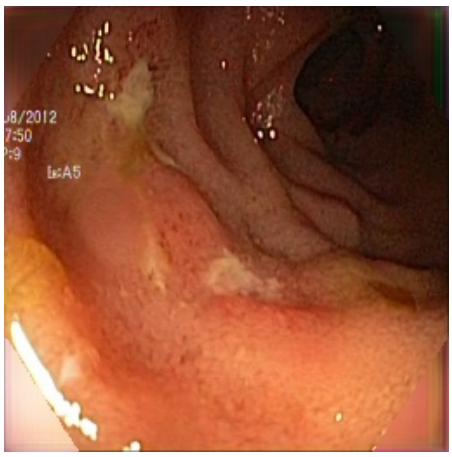
(b) Image from the polyp class at 256x256 px, generated by a GAN .

(c) Image from the normal-z-line class at 256x256 px, generated by an autoencoder

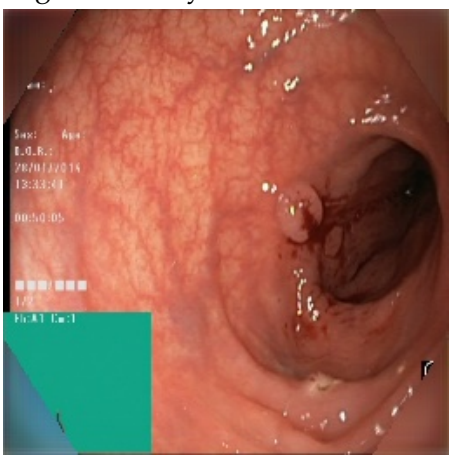
(d) Image from the normal-z-line class at 256x256 px, generated by a GAN

Figure 4.3: Two images from the polyp class and the z-line class. Images on the left were inpained with an autoencoder, and the images on the right were inpainted with a GAN

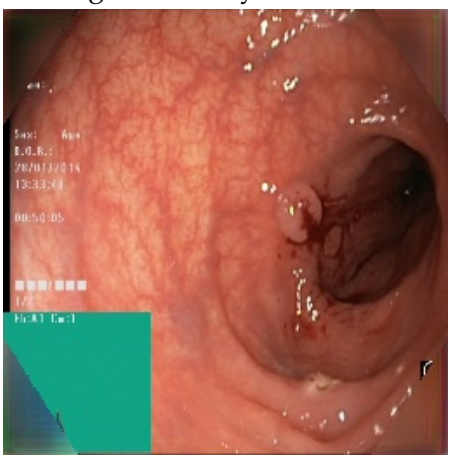
(a) Image from the ulcerative colitis class at 256x256 px, generated by an autoencoder



(b) Image from the ulcerative colitis class at 256x256 px, generated by a GAN



(c) Image from the polyp class at 256x256 px, generated by an autoencoder



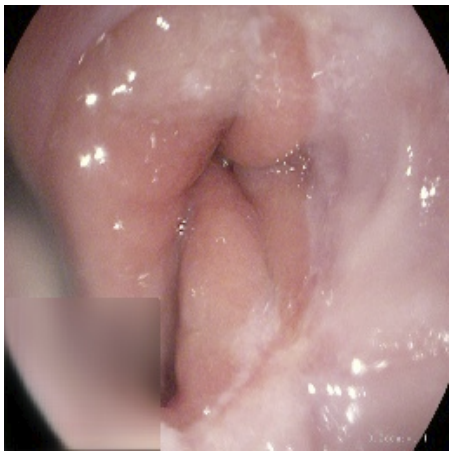
(d) Image from the polyp class at 256x256 px, generated by a GAN

.

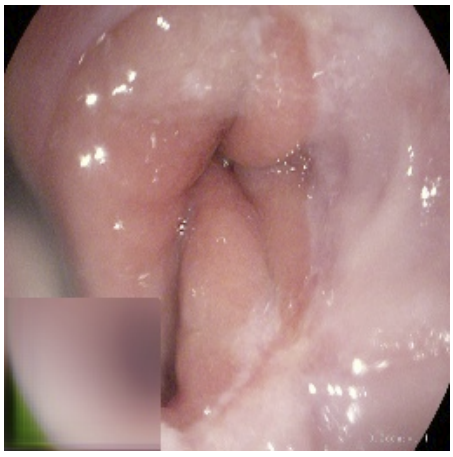
Figure 4.4: Two images from the polyp class and the ulcerative colitis. Here we see results that are not up to a good standard with regards to to light and to green colours.

(a) Image from the polyp class at 256x256 px, generated by an autoencoder

(b) Image from the polyp class at 256x256 px, generated by a GAN .

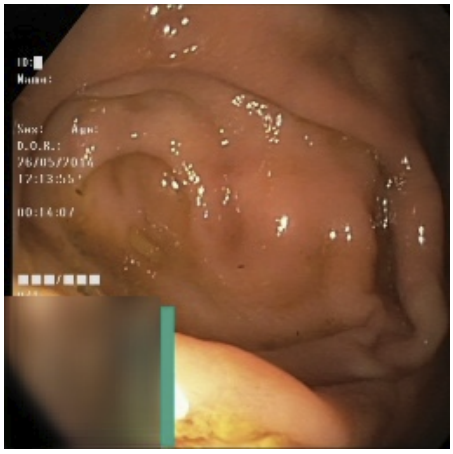


(c) Image from the normal-z-line class at 256x256 px, generated by an autoencoder



(d) Image from the normal-z-line class at 256x256 px, generated by a GAN

Figure 4.5: New text here



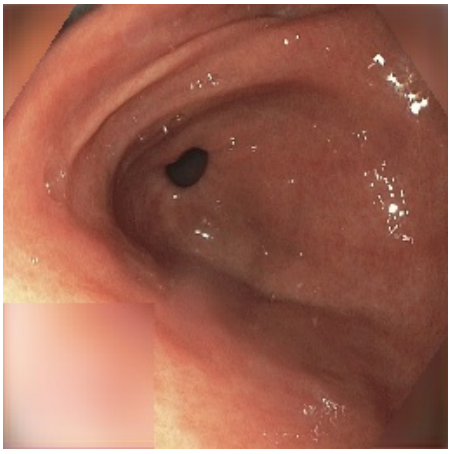
(a) Image from the polyp class at 256x256 px, generated by an autoencoder

(b) Image from the polyp class at 256x256 px, generated by a GAN .

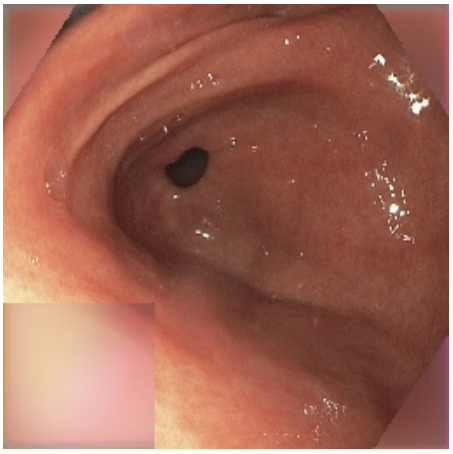
(c) Image from the normal-z-line class at 256x256 px, generated by an autoencoder

(d) Image from the normal-z-line class at 256x256 px, generated by a GAN

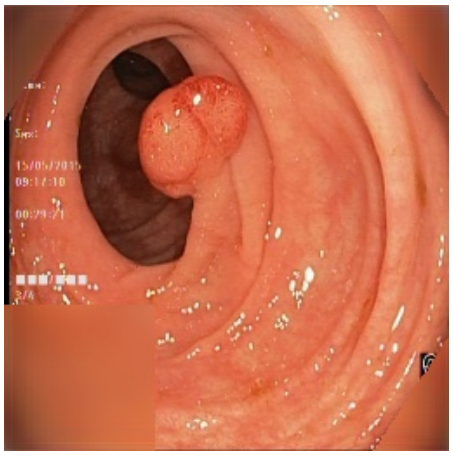
Figure 4.6: New text here



(a) Image from the polyp class at 256x256 px, generated by an autoencoder



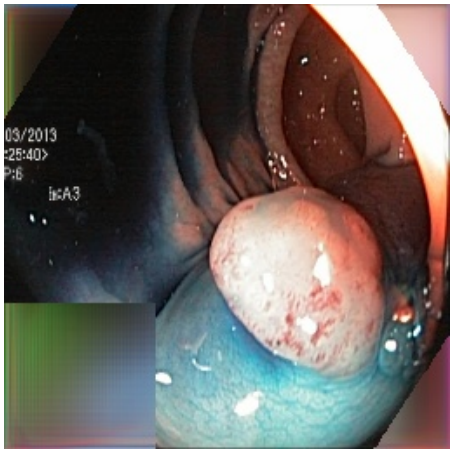
(b) Image from the polyp class at 256x256 px, generated by a GAN .



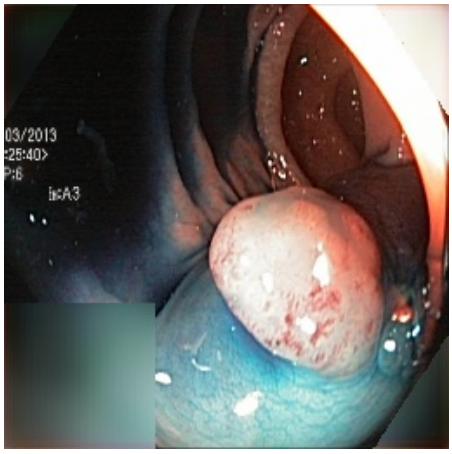
(c) Image from the polyp class at 256x256 px, generated by an autoencoder

(d) Image from the polyp class at 256x256 px, generated by a GAN

Figure 4.7: New text here



(a) Image from the dye lifted polyp class at 256x256 px, generated by an autoencoder



(b) Image from the dye lifted polyp class at 256x256 px, generated by a GAN

(c) Image from the normal-z-line class at 256x256 px, generated by an autoencoder

(d) Image from the normal-z-line class at 256x256 px, generated by a GAN

Figure 4.8: New text here

4.5 Results of the transfer learning experiments

In the last chapter, we looked at the generation of the new datasets. *we looked at the parameters behind the creation of each of them*, and we discussed downfall and advantages with the autoencoder method versus the GAN method.

We are now looking at the results when using the newly created datasets when training a classifier for each one of them.

The model used for the classifier is the same for all the datasets. We use the same learning rate, the same model, and the same parameters for early stopping of the training. In this section we will first go through the model, then we will look at the results for each of the runs.

4.5.1 models

The clssification models was as follows. First, we supply info about the model. This information is for instance batch size, type of network, and choice of an optimiser.

The network chosen is loaded with the imagenet weights. With the model loaded into memory, a global average pooling layer is added, followed by a fully connected layer with the number of nodes equal to the number of classes in the input dataset and a softmax activation step.

With the model loaded we complete it by adding the desired optimiser, and set parameters like batch size, learning rate, validation patience, and image size.

here we talk about K-fold

4.5.2 Base case

When we are looking at the models, we use the base case results as a reference point improve upon. This base case is the result of a standard run of classification without any inpainting.

ref matrix

Matrix shows the evaluation of the test set on each of the three datasets used. On the Kvasir dataset, we used five folds for training and validation, and the last sixth fold for the testing. That left us with 1342 images divided evenly between the classes. In this Confusion matrix, we see that most of the samples lie throughout the diagonal, which, as we recall, is the correct classification.

For the CVC 12k results we have wrong.

Here we can see that the oesophagus and z-line sucks.

We in general need models that do this.

refdatasets

Next, we are looking at CVC12k. As described in chapter we only have 2 classes; polyp and non-polyp.

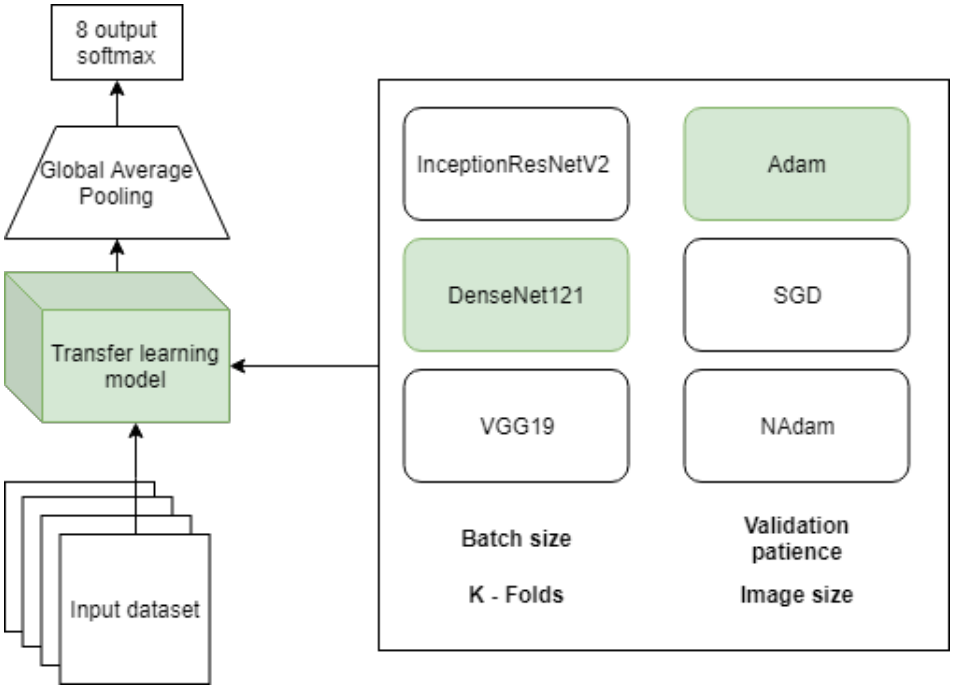


Figure 4.9: The model we use for classifying with the most important options for the learning process.

4.5.3 Corners inpainted result

This dataset is our first dataset, connected to our first hypothesis. Here the corners were inpainted only. From our hypothesis, we would expect the best result to come from the Kvasir test set since the area we remove is not just dataset specific for Kvasir.

This is a table for the AE and GAN with corners.
The first model is the Kvasir dataset.
MCC ACC SPEC
Confusion tables?

Figure shows us the results when training and testing on the Kvasir dataset ref with only corners inpainted.

$$\begin{bmatrix} 1564 & 4362 \\ 365 & 5663 \end{bmatrix}$$

(a) 12k result

$$\begin{bmatrix} 157 & 9 & 0 & 0 & 0 & 0 & 2 & 0 \\ 7 & 157 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 145 & 0 & 1 & 27 & 0 & 0 \\ 0 & 0 & 0 & 158 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 161 & 0 & 2 & 1 \\ 0 & 0 & 21 & 0 & 3 & 139 & 0 & 0 \\ 2 & 0 & 0 & 4 & 0 & 0 & 158 & 0 \\ 0 & 0 & 0 & 4 & 1 & 0 & 4 & 162 \end{bmatrix}$$

(b) The kvasir CM

$$\begin{bmatrix} 1563 & 38 \\ 365 & 318 \end{bmatrix}$$

(c) cvc356 result

Figure 4.10: The baseline confusion matrixes

Kvasir dataset	
MCC	0.9
Accuracy	1
Recall	0.5
Precicion	0.2143
Accuracy	0.1234
F1	0.1234

(a) The kvasir CM

CVC356 dataset	
MCC	0.9
Accuracy	1
Recall	0.5
Precicion	0.2143
Accuracy	0.1234
F1	0.1234

(b) 12k result

CVC 12k dataset	
MCC	0.9
Accuracy	1
Recall	0.5
Precicion	0.2143
Accuracy	0.1234
F1	0.1234

(c) cvc356 result

Figure 4.11: The baseline confusion matrixes

4.5.4 double image size

4.5.5 Inceptionresnetv2

4.6 Classification results

From the results, we see that inpainting the green square gives a better MCC score, compared to the base case When dealing with the images at 256 resolution.

4.7 Summary

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