

# Prostate Cancer Classification using Convolutional Neural Networks

Anna Gummesson

Master's thesis  
2016:E40



**LUND UNIVERSITY**

Faculty of Engineering  
Centre for Mathematical Sciences  
Mathematics

MASTER'S THESIS

# Prostate Cancer Classification using Convolutional Neural Networks

ANNA GUMMESON

Supervisor: Kalle Åström, Centre for Mathematical Sciences

Co-supervisor: Mattias Ohlsson, Department of Theoretical Physics

Examiner: Anders Heyden, Centre for Mathematical Sciences



**LUNDS**  
UNIVERSITET

Centre for Mathematical Sciences  
LUND INSTITUTE OF TECHNOLOGY, LUND UNIVERSITY  
Lund, Sweden 2016



## Abstract

In 2012 prostate cancer was the second most common cancer diagnose for men. The diagnosis is confirmed by pathologists doing ocular inspection of prostate biopsies and the specimens are classified according to the Gleason grading system. The main goal of this thesis is to automate the classification using Convolutional Neural Networks (CNN).

With the introduction of Convolutional Neural Networks the field of pattern recognition broadened. The classical way of designing and extracting hand-made features for classification is substantially different to letting the computer itself decide which features are of importance, the new approach was enabled by CNNs. This together with groundbreaking results on benchmark image sets has made CNNs a well-used method in pattern recognition.

In this thesis a CNN with small convolutional filters has been trained from scratch using stochastic gradient descent with momentum. The error rate for the CNN is 7.3%, which is significantly better than previous works using the same data set. Since good results were obtained even though the data set were rather small, the conclusion is that CNNs are a promising method for this problem.

Keywords: ANN, CNN, deep learning, prostate cancer classification, automated Gleason grading.



## Acknowledgements

First I want to thank my supervisors Kalle Åström and Mattias Ohlsson who have with much dedication guided me through this. Secondly I want to thank Roy Ehrnström and Felicia-Elena Marginean at Skåne University Hospital who has taken time to explain the pictures to me and showing me that the diagnostics are far more complex than I thought when I first encountered the images. Agnieszka Krzyzanowska also has my thanks for providing material and being an excellent interpreter between engineers and pathologists.

Last I want to thank those that wrote their master thesis at the center for mathematical sciences alongside with me. Thanks for good company, and in times of adversity the odd fika-discussions and lunches really brightened my day.

Anna Gummeson, Lund, August 2016



# Contents

<b>Glossary</b>	<b>ix</b>
<b>Acronyms</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Aim of the thesis . . . . .	1
1.2 Related work . . . . .	2
1.3 Structure of the report . . . . .	2
<b>2 The Theory of Artificial Neural Networks</b>	<b>5</b>
2.1 Classification . . . . .	6
2.2 The simple perceptron . . . . .	7
2.2.1 Activation function . . . . .	7
2.3 Multilayer Perceptron . . . . .	8
2.4 Convolutional Neural Networks . . . . .	9
2.4.1 Structure . . . . .	9
2.4.1.1 Convolutional layer . . . . .	9
2.4.1.2 Pooling layer . . . . .	12
2.4.1.3 MLP . . . . .	12
2.4.1.4 Softmax . . . . .	12
2.4.2 Performance on other data sets . . . . .	13
2.5 Training a neural network . . . . .	13
2.5.1 Error function . . . . .	13
2.5.2 Stochastic gradient descent with momentum . . . . .	13
2.5.2.1 Gradient descent . . . . .	13
2.5.2.2 Mini-batch update . . . . .	14
2.5.2.3 Stochastic gradient descent . . . . .	14
2.5.2.4 Stochastic gradient descent with momentum . . . . .	14
2.6 Generalisation methods . . . . .	14
2.6.0.1 Weight decay . . . . .	15
2.6.1 Expand data set . . . . .	15
2.6.2 Dropout . . . . .	16
<b>3 Gleason Grading</b>	<b>17</b>
<b>4 Dataset and Materials</b>	<b>19</b>



4.1	Images . . . . .	19
4.1.1	Ethics . . . . .	19
4.2	Software . . . . .	19
<b>5</b>	<b>Methods</b>	<b>21</b>
5.1	K-Fold Cross-Validation . . . . .	21
5.2	Pre-processing . . . . .	21
5.3	Parameters and net design . . . . .	22
5.3.1	Convolutional and max-pooling blocks . . . . .	22
5.3.2	MLP . . . . .	23
5.3.3	Learning rate . . . . .	23
<b>6</b>	<b>Visualization of detected features</b>	<b>25</b>
<b>7</b>	<b>Results</b>	<b>29</b>
7.1	Misclassified images . . . . .	30
<b>8</b>	<b>Discussion</b>	<b>37</b>
8.1	Future work . . . . .	38
	<b>Bibliography</b>	<b>39</b>

# Glossary

- batch** A set of patches/inputs. 14
- epoch** An epoch has passed when all batches has been run through the learning algorithm. 30
- filter** Parts of a convolutional layer, the layer input is convoluted with the different filters as to produce the layer output. 9
- Gleason score** Standardized diagnostic tool for classifying prostate cancer. 18
- image** Original fullsize image. 10
- layer** The building blocks of an ANN. There are different kinds, such as convolutional, max-pooling, softmax etc. 8
- patch** A down sampled cutout from an image of fixed size. Used as input to a neural network. 14



# Acronyms

<b>ANN</b>	Artificial Neural Network.	5
<b>CNN</b>	Convolutional Neural Network.	9
<b>GD</b>	Gradient descent.	13
<b>MLP</b>	Multilayer Perceptron.	8
<b>PC</b>	prostate cancer.	1
<b>ReLU</b>	Rectified Linear Unit.	7
<b>SGD</b>	Stochastic Gradient Descent.	14



# 1

## Introduction

In 2012 prostate cancer (PC) was the second most common cancer diagnosis for men, cf. [1]. For correct treatment of these patients, a good classification of the severity of the cancer is necessary. The most used method to do this classification is to give each specimen a Gleason score. The task of grading the samples is done by ocular inspection and is time consuming work. Even though only a few seconds are spent per image, the workload is heavy since prostate cancer is very common. Also we have today a lack of pathologist in Sweden, resulting in long queues for cancer patients, cf. [2]. Due to this there is good reason to try to automate the diagnosis process.

Another motivation is that it has been shown in [3] that different pathologists grade differently. With a second opinion from a program, individual diagnoses might come closer to consensus, preventing over- and under-treatment.

The introduction of CNNs has broadened the field of pattern recognition. It replaces the classical way of designing and extracting hand-made features for classification with the substantially different strategy of letting the computer itself decide which features are of importance. Ground-breaking results on benchmark image sets has made CNNs a well-used method in pattern recognition.

### 1.1 Aim of the thesis

The aim of this thesis is to implement a classifier for microscopic images of potentially cancerous tissue from prostates using a Convolutional Neural Network. This thesis is a pre-study to a larger research project in collaboration with Skånes Universitetssjukus (SUS) and SECTRA.

### 1.2 Related work

Previous work on automated classification of Gleason grading include Gorelick et al. [4] who first segment the microscopic images into pathologically meaningful segments (stroma, lumen, lymphocyte etc.) using superpixels. The segmentation is thereafter used for classification. In Doyle et al. [5] a Bayesian multi-resolution approach using AdaBoost is proposed for classification. Lippolis [6] uses a method based on SIFT-points. These are then clustered and each image is represented of the corresponding histogram. The histograms are then used as feature vectors when classifying the images. An example when a CNN has been used is Källén et al. [7], who used a pre-trained convolutional network for feature extraction, and then a Support Vector Machine (SVM) or Random Forest for classification. The two last works have used the same data set as is used in this thesis, the results are therefore somewhat comparable. A very interesting and recent article is Litjens et al. [8], who have tailored a CNN to classify hematoxylin and eosin stained images of prostatic tissue.

In this thesis we will fit a CNN from scratch to this specific problem. In contrast to Gorelick et al. [4] the detour around segmentation will not be necessary, however CNNs can be used for segmentation of images with good performance. It does though have a more medical foundation than Doyle et al. [5] and Lippolis [6] whom construct features based on image properties more than physiological information. Källén et al. [7] uses a pre-trained convolutional network, a net not designed to only classify prostatic tissue. The expectation is therefore that a specialised network will be smaller, faster and hopefully better. The work in this thesis has similarities with Litjens et al. [8]. One of the main differences are that they are only interested in classifying cancer/not cancer but this thesis aims to separate the cancerous images into Gleason score as well. Another distinction is that they have used many more images, which they have produced themselves. Using few examples to train on and tackle the problem of stain invariance between hospitals are problems addressed in this project.

### 1.3 Structure of the report

There are three main parts of this thesis. First, in chapter 2 and 3, we will briefly go through theory. First Artificial Neural Networks is explained, leading to Convolutional Neural Networks, which is the foundation of this project. In chapter 3 you will find a brief explanation on Gleason grading, which the classification is based on. The second part, chapter 4 and 5, describes my setting. First presenting the images and programmes used, and in chapter 5 how I have tried to solve the classification problem.

The last part contains the result, chapter 7, and comments and analysis on these in chapter 8. Chapter 6 is a side project, trying to present a tool for visualizing feature extraction performed by the network.

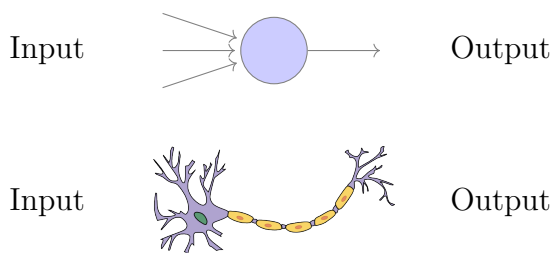




# 2

## The Theory of Artificial Neural Networks

An Artificial Neural Network (ANN) is a construction inspired by the human brain, as the word *neural* indicates. There are similarities between a nerve cell and a node belonging to an ANN. A nerve cell have dendrites handling the cell input and if the stimuli is large enough the signal is passed on to new cells through axons. ANNs are constructed of nodes with weighted input, an activation function and output. A comparison between a neuron and its counterpart in Artificial Neural Networks can be seen in figure 2.1.



**Figure 2.1:** Comparison between neuron and its counterpart in Artificial Neural Networks.

In [9] an attempt to define ANN is: "*A neural network is a massively parallel distributed processor made up of simple processing units that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:*

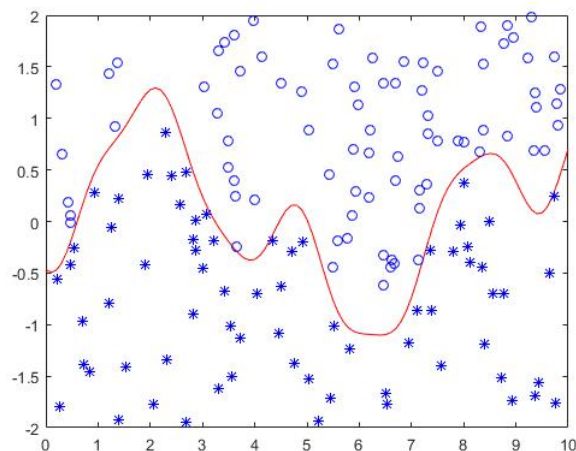
1. *Knowledge is acquired by the network from its environment through a learning process.*
2. *Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge."*

Due to ANNs capacity and black box behaviour it is easy to think of it as complicated. Large complex structures may be built with amazing results, but the smallest elements are rather simple. In this section we shall start with the simplest possible network and successively build larger and more complex structures, ending with CNNs, which have been used in this project.

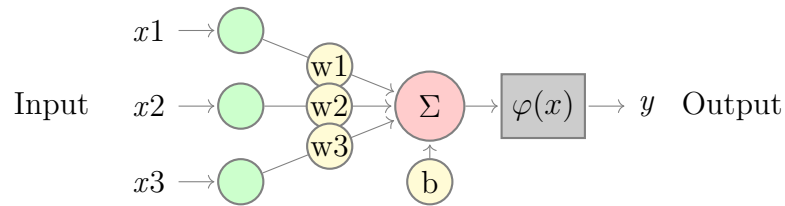
## 2.1 Classification

A brief introduction to the problem of classification is suitable before we explain ANNs. In a classification problem we have data points  $x$  and their corresponding labels  $d$ , the label tells us which class  $x$  belongs to. For example see figure 2.2. It shows a binary classification problem where the inputs  $(x_1, x_2)$  are the coordinates and the circles and asterisks marks the class they belongs to. The red line is called a decision boundary and is the result of a classifier. New data points presented to the classifier will be classified as one class on one side of the boundary and another class on the other side. The aim of classification algorithms is to find good decision boundaries. ANNs could be used in more applications than classification problems, but those will not be covered in this thesis.

If ANNs are used for classification problems it is common to interpret the outputs as probabilities of the inputs belonging to each class. If we have the example presented before, circles mark class 0 and asterisks mark class 1. The output from the ANN is  $y_0$  and  $y_1$ , which sums to 1. The decision boundary is where  $(y_0, y_1)$  is constant, often  $(0.5, 0.5)$  is chosen. This means that when  $y_0 > y_1$  the algorithm will classify the data point as 0 and vice versa. In this thesis inputs are  $106 \times 106$  images and the outputs are  $(y_B, y_{G3}, y_{G4}, y_{G5})$ , since we use four classes.



**Figure 2.2:** An example of a classification problem in two dimensions. Decision boundary is marked in red.



**Figure 2.3:** Schematic representation of a simple perceptron.

## 2.2 The simple perceptron

The simple perceptron is the smallest possible ANN, it consists of only one neuron. The schematic representation can be seen in figure 2.3.

In this figure we see input variables  $x_i$ , corresponding weights  $w_i$ , the weighted sum of the inputs, an activation function  $\varphi$ , the output  $y$  and the bias  $b$ . Bias is an additional input to each neuron often represented as an input  $x = 1$  and a weight  $w_0$  ( $w_0 = b$ ). Unlike other applications bias is an often necessary thing in ANNs.

The output  $y$  from the single perceptron is

$$y = \varphi\left(\sum_{k=1}^m \omega_k x_k + b\right).$$

Including the bias in the summation we get

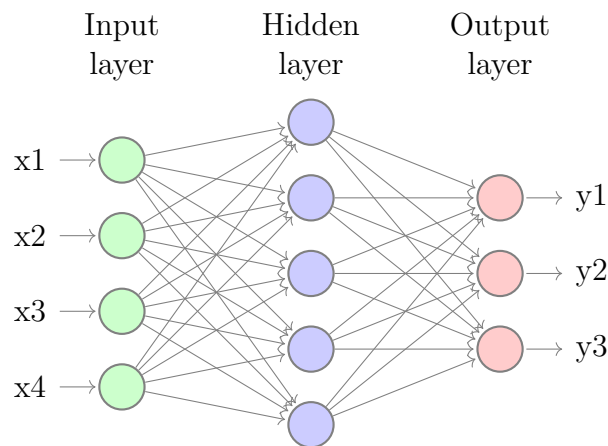
$$y = \varphi\left(\sum_{k=0}^m \omega_k x_k\right).$$

Using a single perceptron the decision boundary is a hyperplane since we have linearity, (cf. linear SVM). With a single perceptron we may for example solve the AND and OR problem.

### 2.2.1 Activation function

$\varphi$  in figure 2.3 is called an activation function. For regular ANNs a sigmoid function is often used, as the logistic function or  $\tan(x)$ . Practice is to use the Rectified Linear Unit (ReLU) when working with CNNs. It is defined as

$$\varphi(x) = \begin{cases} 0 & \text{if } x \leq 0, \\ x & \text{if } x > 0. \end{cases}$$



**Figure 2.4:** Schematic representation of a Multilayer Perceptron.

## 2.3 Multilayer Perceptron

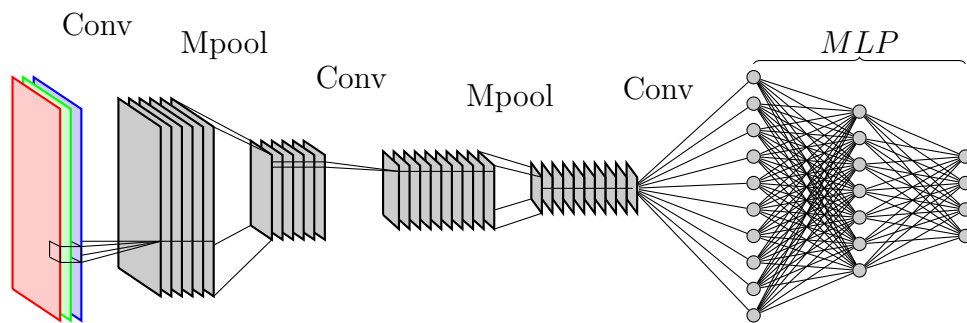
Connecting many neurons we may construct a web, where one neurons output may be another neurons input. This may be done arbitrary, but this thesis will only work with so called feed-forward networks, ANNs that have no feed-back, i.e. input comes in from one side, propagates through the net and comes out as output at the other end. Such a network is called a Multilayer Perceptron (MLP), each step forward is called a new layer, and each layer is constructed of perceptrons. The term Deep Neural Networks or deep learning are also used, this originates from that adding layers increases the depth of the network.

Say we have a classification problem of three classes and a net looking like the one in figure 2.4. Then the largest output  $y_i$  is the most probable class for the input  $\mathbf{x}$ . The function for calculating these outputs is

$$y_i = \varphi_o\left(\sum_{j=0}^5 w_{ij}\varphi_h\left(\sum_{k=0}^4 \tilde{\omega}_{jk}x_k\right)\right)$$

as may be deduced from the graph. In classification it is common to use softmax on the output to get  $y_i$  as probability for the input  $\mathbf{x}$  to belong to class  $i$ . Softmax will be explained in section 2.4.1.4.

When adding hidden layers the decision boundary is no longer limited to a hyperplane but can result in non-linear solutions.



**Figure 2.5:** Schematic representation of a Convolutional Neural Network.

## 2.4 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a deep neural network that take account for spatial dependence in the input data, they were first introduced in [10]. Due to further development, both of theoretical nature (as dropout c.f. [11]) and computational sort (the use of GPUs), the use of CNNs is today well-spread. Because of the spatial dependence they are prominent in image analysis applications.

### 2.4.1 Structure

In a Convolutional Neural Network the hidden layers have different purposes and different constellations. Most of these layers are optional, but there has to be at least one convolutional layer for the net to be called a convolutional network. Another difference is that the hidden nodes are not a single neuron anymore, but a filter of spatially fixed neurons. The type of Convolutional Neural Network for classification used in this thesis takes images as inputs and present the probability of the input image belonging to each class as output. The input image is of fixed size, so one solution when the images are of different sizes is to use equisized cut outs called patches. A schematic representation of a Convolutional Neural Network can be seen in figure 2.5, and below you will find explanations of the different parts.

#### 2.4.1.1 Convolutional layer

As the name implies convolution is a major part of the function of these networks. Convolution of a function  $x$  with a kernel  $w$  in one dimension is defined as

$$s(t) = \int x(a)w(t - a)da,$$

or as a discrete function

$$s(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a),$$

i.e a flip and summation over the kernel. Extending this to two dimensions results in

$$S(i, j) = \sum_m \sum_n I(m, n)F(i-m, j-n),$$

i.e. flip the filter (F) up-down and left-right and sum over all products.

An example can be seen in the top picture in figure 2.6. To view the calculation we will take the marked pixels as an example:

$$\text{Input image} = \begin{bmatrix} 9 & 6 \\ 0 & 1 \end{bmatrix}, \quad \text{Double flipped filter} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

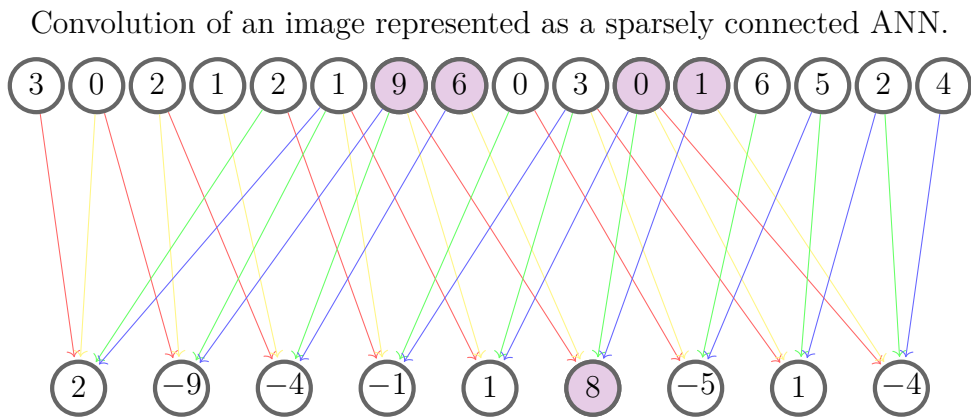
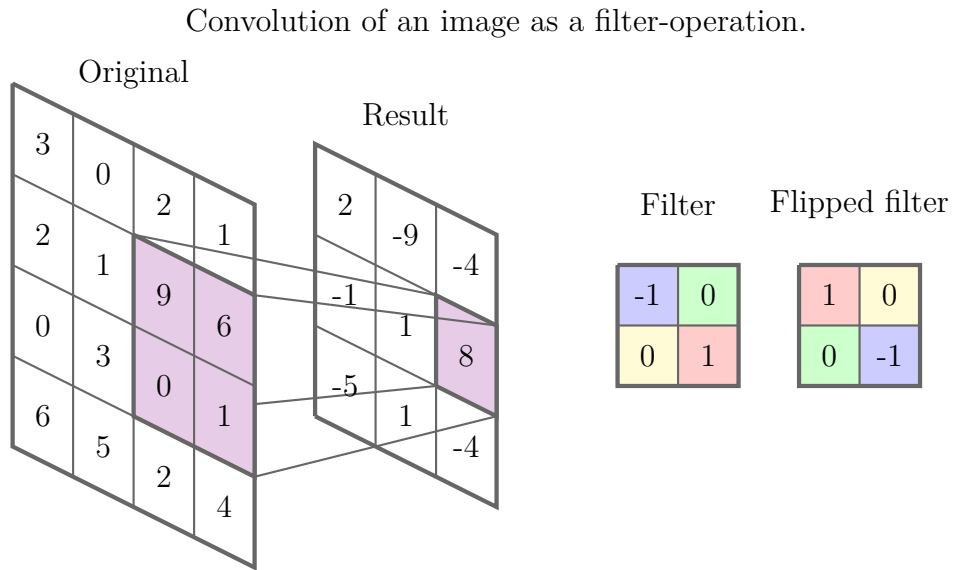
Result of the convolution at this point is then

$$1 \cdot 9 + 0 \cdot 6 + 0 \cdot 0 + (-1) \cdot 1 = 8.$$

How this translates into an ANN can be seen in the bottom image in the same figure. There are two peculiarities to be noticed about this network. Firstly it is not fully connected, all neurons in a layer do not have contact with all nodes in the previous layer. A net like this is called sparsely connected. Secondly we have weight sharing, all connections of the same color has the same weight, namely the values of the corresponding pixel in the filter.

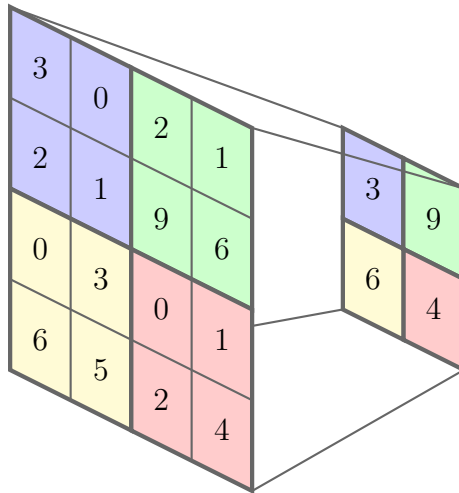
These two features gives the convolutional layer its characteristics, that they are hard to overtrain, since we have much fewer weights to train than the possible connections, and that they introduce a spatial invariance. This means that if the subject is to detect a dog in the image, it should not matter if the dog is in the center or in a corner. The convolutional layer ends with an activation function, in our case the Rectified Linear Unit (ReLU).

In a convolutional layer several of these filters are used. For example the first convolutional layer in figure 2.5 takes an image with three colour channels and outputs a new image with six "channels". For this we need 18 filters, one from each input channel to each output channel.



**Figure 2.6:** An example showing how convolution of an image works, both traditionally and as a neural network.





**Figure 2.7:** The principle of max-pooling.

### 2.4.1.2 Pooling layer

Pooling layers cause a down-sampling of the filter outputs. This thesis has used max-pooling with a filter size of  $2 \times 2$  and stride 2. It means that the largest value within this filter is saved to the next layer and that instead of moving one step to do the next calculation as in the convolutional layer we move two steps. See example in figure 2.7. Thus no pixel is tested for largest value twice and 75% of the information in the image is discarded. Other types of pooling is available, for example avg-pooling, saving the average of the inputs instead of the maximum. The pooling is done separately for all images, so the number of channels does not change from the previous layer.

### 2.4.1.3 MLP

It is common to finish a CNN with a Multilayer Perceptron. In practice this is just convolutional layers with only one pixel per filter.

### 2.4.1.4 Softmax

The output from the network can be hard to interpret. In classification problems it is common to finish the CNN with a softmax function

$$\tilde{y}_i = \frac{e^{y_i}}{\sum_j e^{y_j}}.$$

This normalisation forces the sum of outputs to one and therefore the softmax output  $\tilde{y}_i$  can be interpreted as the probability of the input belonging to class  $i$ .

### 2.4.2 Performance on other data sets

One major breakthrough for CNNs was when [12] achieved outstanding results on the ImageNet 2012 data set (1.2 million images of 1000 classes). Since then CNNs have been successfully applied to several image benchmark set, including the renowned MNIST database (70 000 handwritten digits), where they hold the record to date [13]. CNNs ought to work very well for classification of prostatic tissue since it is a similar problem.

## 2.5 Training a neural network

Before the learning process begins some parameters are fixed as which activation functions to use, what kind of layers to use and how many. The training of an Artificial Neural Network is a process of updating its weights. This is done using a method called backpropagation.

### 2.5.1 Error function

In classification problems every input  $x(n)$  has a label, a desired output  $d(n)$ . We may then construct an error function as

$$E(w) = - \sum_{n=1}^N \ln \left( \frac{e^{y_{d(n)}(n)}}{\sum_{i=1}^k e^{y_i(n)}} \right).$$

This function is called the cross-entropy error and is commonly used for classification problems since it has high penalties for miss-classifications.  $n$  is the input pattern tested and  $i$  is the index for the output node, i.e. in our case  $k = 4$ . The objective of the training algorithm is to minimize this function.

### 2.5.2 Stochastic gradient descent with momentum

To minimize the error function we use an algorithm called stochastic gradient descent with momentum. We will build this method in steps.

#### 2.5.2.1 Gradient descent

Gradient descent (GD) means that we take a unit step in the direction of the greatest descent as

$$w_{jk\ new} = w_{jk} + \Delta w_{jk}, \quad \text{where } \Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}}.$$

Here  $\eta$  is the step size which is commonly denoted learning rate in ANN context. For a discussion on how the learning rate affects the optimization see fig. 5.2. To update the weights after all patches has been considered is called batch update.

### 2.5.2.2 Mini-batch update

Instead of updating the weights after all patches  $n$ , we may split them into equisized mini-batches and do the update with respect to all patches in each mini-batch as

$$\Delta w_{jk} = -\eta \sum_i \frac{\partial E}{\partial w_{jk}}.$$

### 2.5.2.3 Stochastic gradient descent

Stochastic Gradient Descent (SGD), with its modifications, is a very common algorithm for machine learning, and deep neural networks is no exception. The method means that not all patches in each mini-batch is used for update but only a random sub-sample  $i \in I$  as

$$\Delta w_{jk} = -\frac{\eta}{|I|} \sum_{i \in I} \frac{\partial E}{\partial w_{jk}}.$$

### 2.5.2.4 Stochastic gradient descent with momentum

Stochastic gradient descent with momentum works as adding a moving average to the SGD, this is done with

$$\Delta w_{jk}(t+1) = -\frac{\eta}{|I|} \sum_{i \in I} \frac{\partial E}{\partial w_{jk}} + \alpha \Delta w_{jk}(t).$$

The momentum term can be interpreted as dropping a ball in the error landscape. When moving in approximately the same direction for many steps it can roll "uphill" for a bit, possibly avoiding bad local minima. This update method is considered to be faster than SGD. The parameter  $\alpha$  determines how fast previous gradients decay.

## 2.6 Generalisation methods

Generalisation is the ability for a model to work well on input that was not in the data set used to train the model. The generalisation is often measured by holding out a validation set which is not included in the training data. By analysing the error made by the net on the validation data we get an estimation of how the net will perform on unknown data.

The most common explanation to the situation when we have close to zero training error (error on the examples used for training) but a much larger validation error is overfitting. Overfitting or overtraining is when there is more complexity in the model than necessary, this means that instead of learning the concept and having a good algorithm the net tries

to exactly remember the input data. For comparison, it is as learning

$$[0, 2] \rightarrow \text{class 2}, \quad [1, 1] \rightarrow \text{class 2}, \quad [2, 1] \rightarrow \text{class 3}, \text{ etc.}$$

for every number pair separately instead of learning the summation operation. Overtraining can also be interpreted as the network adjusting to noise in the data instead of just the underlying function. This problem could be addressed by reducing the complexity of the network, i.e. reduce the number of weights used. The effect will be that less parameters will be available to adjust to the noise, these methods are called regularisation techniques. In this thesis weight decay, expansion of data set and dropout have been used to prevent overtraining and improve generalization.

### 2.6.0.1 Weight decay

Weight decay is a regularisation technique which add a term to the error function as

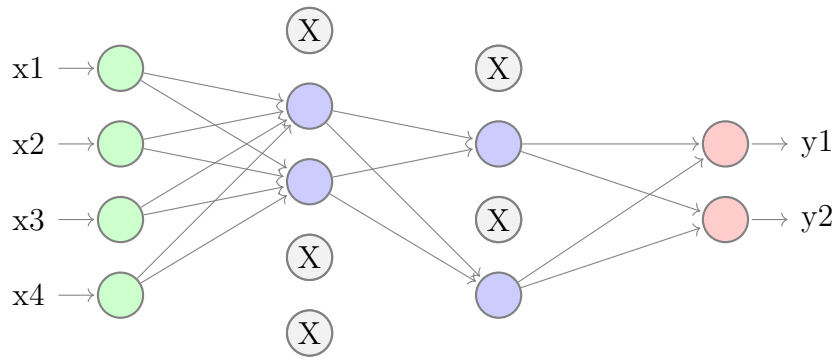
$$E(w) = - \sum_{n=1}^N \log \left( \frac{e^{y_{d(n)}(n)}}{\sum_{i=1}^k e^{y_i(n)}} \right) + \frac{\lambda}{2} \sum_l w_l^2.$$

By this alteration we are forcing superfluous weights to zero and therefore preventing overfitting.

## 2.6.1 Expand data set

The more data available for the network the less noise will be accounted for and the generalisation will improve. Often more data is either hard or impossible to attain so one solution is to artificially expand the data set. Usual practice includes adding noise, enlarge or shrink the images slightly, rotating, skew the images etc. and adding these altered images to the data set. Using these methods the input data set may be substantially larger. Which methods are reasonable to use is problem dependent. If the images contain characters a slight scale adjustment, rotation or skewness might work but flipping the image upside down or left to right would be a bad idea since the representation of most of the characters change.

Microscopic images of prostatic tissue is invariant of rotations and flipping. In this thesis patches were extracted from the original images, the images then rotated ten degrees and new patches were extracted, this process was iterated for the full 360 degrees and the same procedure was performed for the flipped images. The size of the original image decides on how many patches were extracted, this ranges from none to several hundreds. When no patches were extracted the patch-size has been larger than either the length or height of the image.



**Figure 2.8:** Schematic representation of the dropout regularisation technique.

### 2.6.2 Dropout

Dropout is a regularisation technique first introduced in [11]. Before presenting a new input to the net during training the individual nodes are excluded from the net with a probability  $p$ . The output nodes are all kept and it is also common to keep all the input nodes. A figure to explain the exclusion is shown in 2.8.

Since each neuron cannot count on specific others to be present they all have to represent more general features, which results in that complex co-adaptations are prevented. Dropout also works as an ensemble averaging algorithm. Instead of training several independent networks and weighting together the result to reduce variance in the estimation, dropout have the same tendencies but with only one, bigger, network.

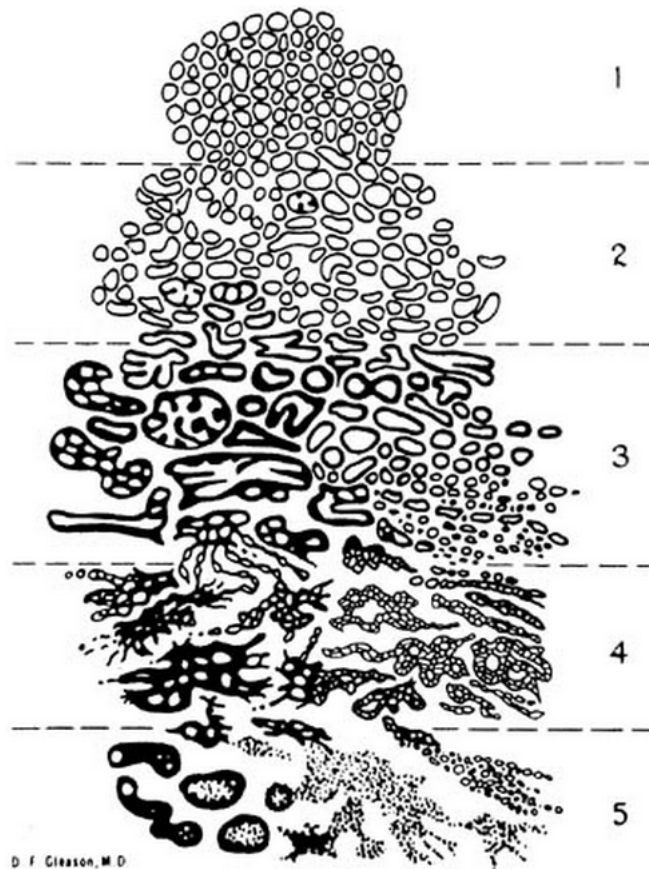
In CNNs dropout is not as important to implement as in other ANNs since the shared weights in the convolutional layers already forces the filters to represent more general features and therefore has a natural resistance to overfitting. In this thesis dropout is only applied to the concluding MLP with  $p = 0.5$ .

# 3

## Gleason Grading

When prostate cancer is discussed it is often implied that the subject is adenocarcinoma, cancer that affects the glands. Benign tissue consist of well-defined glands with basal cells present. They are often branched with papillary infolding. Cancerous tissue lacks basal cells, typically have smaller glands and the nuclei are large and may contain one ore more large nucleoli, cf. [14].

Gleason grading was invented by Donald Gleason in the sixties as a way to describe patterns and configurations in prostate tissue. By looking at the glands one may then measure how aggressive a prostate cancer is. How Gleason described the stages can be seen in figure 3.1.



**Figure 3.1:** Original Gleason grading for prostatic adenocarcinoma.

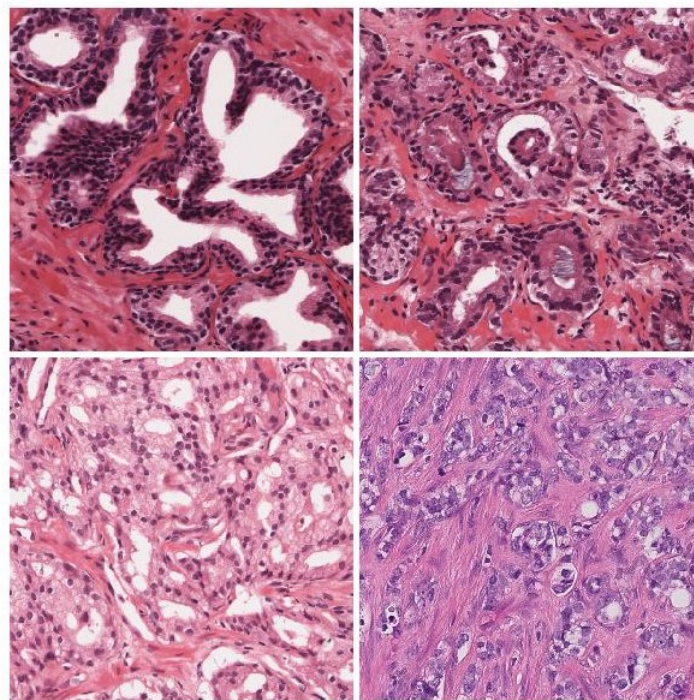
### 3. Gleason Grading

---

Over time the partitions have shifted slightly and only Gleason 3-5 are in pathological use today. A definition of the current Gleason scores is found in [15]:

Score	Characteristics
3	<i>Well-formed glands</i>
4	<i>Poorly formed/fused/cribriform glands</i>
5	<i>Lack of gland formation (or with necrosis)</i>

For examples of the different scores shown in microscopic images see figure 3.2. These images are part of the data set used.



**Figure 3.2:** Representation of healthy tissue and the three stages of cancer. Top left: Benign, top right: Gleason 3, bottom left: Gleason 4, bottom right: Gleason 5.

In this thesis the objective is to classify the Gleason score, i.e. separate the classes benign tissue and Gleason score 3, 4 and 5 respectively.

# 4

## Dataset and Materials

### 4.1 Images

The images used in this thesis come from two sources, Beamount Hospital in Dublin and the Prostate Cancer Research Consortium and PathXL in Belfast. The images picture sliced biopsies of prostatic tissue that has been stained with hematoxylin and eosin, and scanned to produce digital images. The different origins have resulted in slight differences in saturation of the samples.

Each image contains but one class, and differs a lot in size, (from 491 568 - 40 086 725 pixels.) The pictures are taken at 40x magnification. The image set contains 52 images of Benign tissue, 52 images of tissue with Gleason score 3, 52 images of tissue with Gleason score 4 and 57 images of tissue with Gleason score 5. Five of the images were too small to extract patches and were therefore discarded. After data set expansion more than 12 000 patches were extracted for each class. To not emphasize certain classes the same number of patches from each class were used for training.

#### 4.1.1 Ethics

All images are anonymous in the sense that no images can be connected to the patients. As this thesis is part of a larger research project on human tissue an application has been sent to and approved by the Central ethical review board, the ethical permit is 2013/400.

### 4.2 Software

The software that has been used is a package for MATLAB called Mat-ConvNet, developed by the Oxford Visual Geometry Group. The home page for the project can be found at [16].





# 5

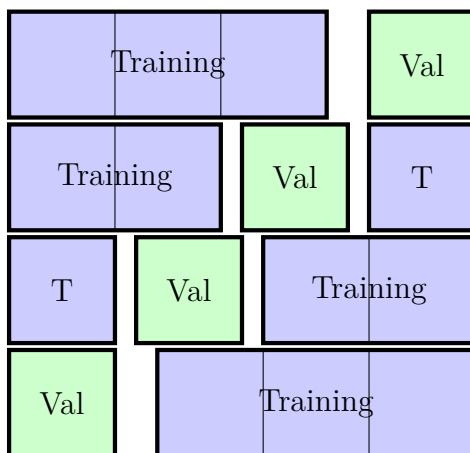
## Methods

### 5.1 K-Fold Cross-Validation

Cross-validation is used to reduce the variance in the validation error estimates without having to set aside a lot of data for validation. The data is split into  $k$  approximately equal partitions, the splitting can be seen in figure 5.1, each of these partitions are then used as validation data in  $k$  different rounds, in this thesis  $k = 4$  was used. The final result is then the mean of the  $k$  results. The images were randomly permuted before the split as to avoid any unwanted dependence due to the images different origins. This is performed before extraction of patches so that patches originating from the same image is never present in training and validation set at the same time.

### 5.2 Pre-processing

No conventional pre-processing was performed, however each color-channel (red, green and blue) was individually normalized between 0 and 255 and the mean was extracted. To normalize the inputs in some way is common practice when using an ANN. For example if we want to estimate the growth of a certain bacteria in a tube, where the input parameters



**Figure 5.1:** The principle of 4-fold cross-validation.

are number of bacteria and amount of nutrition in kg. These numbers will probably differ by several orders of magnitude. Introducing the input to a randomly initialized network will cause the bacteria input to drown the nutrition impact. Since the presence of nutrition can be the difference between exponential growth or decreasing amount of bacteria the parameter is important for good analysis. To handle this problem the weights from the bacteria input must be trained to be small and the nutrition weights must increase. This scheme is both slow to train and the risk is that the net get stuck somewhere along the way, and therefore is normalisation of inputs important.

The images were also down sampled with a factor 0.15. This was mainly due to practicalities, having large images as inputs result in more weights to train, which leads to longer training times. The computer running out of memory is also a real problem when inputs and networks are enlarged.

### 5.3 Parameters and net design

Designing a CNN includes tuning several hyper-parameters. These include what order and of what kind should the layers be, which dimensions should the filters have, how many filters there should be in each convolutional layer, etc. To be able to finish the thesis in the given time frame a few of these hyper-parameters have been fixed.

#### 5.3.1 Convolutional and max-pooling blocks

I have used multiple max-pooling layers in the network. It is an effective way to quickly reduce the size of the filtered images, since 75% of the information is disregarded in each layer if a 2x2 filter is used. This makes the total net shorter and reduces the number of weights used. To use this effect the net design has been built around blocks of one convolutional layer with ReLU added to each output and a max-pooling layer.

As suggested by [17], very small (3x3) filters have been used. Also a few 4x4-filters were included to fit the images without padding the edges when max-pooling was applied. The max-pooling specifications used (2x2 with stride 2) also coincide with what was used in [17].

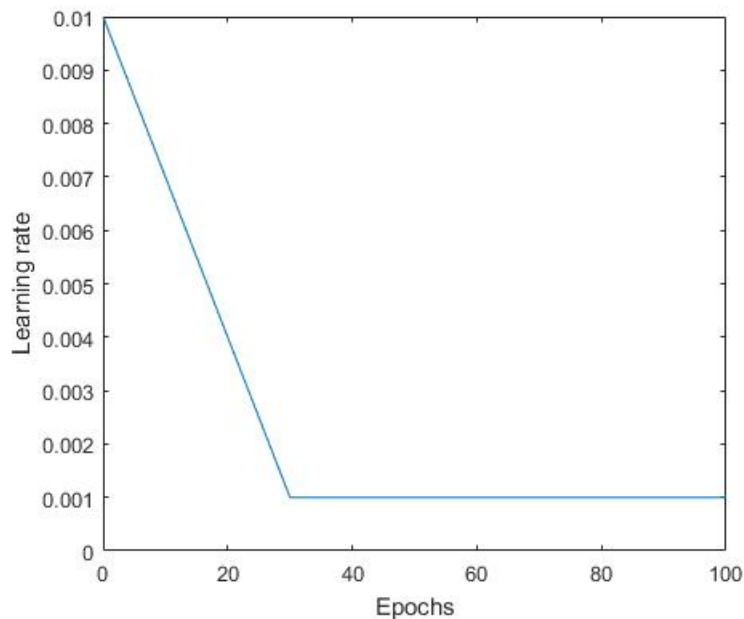
The hyper-parameters most adjusted in this project have been how many filters should be used in each convolutional layer.

### 5.3.2 MLP

This thesis have investigated both using one and two layers of MLP, but the most tuning is to decide how many neurons to use in each layer to allow for enough complexity but prevent overfitting. Experiments with and without dropout in this step has also been performed.

### 5.3.3 Learning rate

According to [18] the learning rate is the most important hyper-parameter to tune. Using a too large learning rate may cause the training algorithm to have problem with convergence, and a too small learning rate may get the algorithm to get stuck in a local minima with bad generalization. For this thesis an adaptive learning rate, large in the beginning and smaller at the end, has been used with the purpose of finding good minimas and reaching the optimum at that minima. For simplicity a piecewise linear function was used, see figure 5.2.



**Figure 5.2:** The learning rate as a function of epochs.



# 6

## Visualization of detected features

This chapter is not a part of the main project, but it is included as a guide to understand the effects and ways of the CNN.

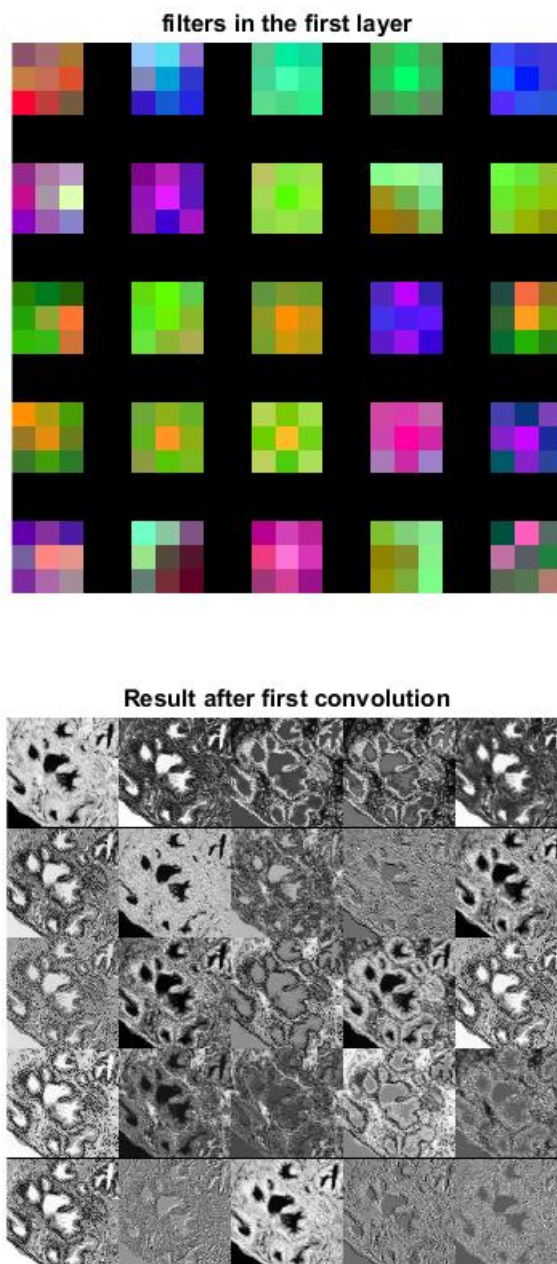
It can be hard to interpret what a CNN is doing. Which features are extracted and in what way they affect the final classification. The black-box behaviour of CNNs makes net design somewhat of a trial-and-error scheme and better understandings of the network might lead to improved networks.

The first layer is easy to interpret just by showing the filters of the first layers and what the images look like after the convolution. This can be seen in figure 6.1. Here we find both low and high-pass filters (edge detectors) both within color layers and between them. Later convolutional filters are harder to plot since we only have three color channels and for example the next convolution will have an input with 25 channels.

There are advanced ways to visualize what a CNN does on a higher level using a deconvnet, see [19]. A much simpler method for feature visualisation has been tested in this thesis which is based on a method by Erhan et al. [20]. They suggest a method they call *Maximizing the activation*. First we choose a unit of interest  $i$  in layer  $j$ , and let  $h_{ij}(\theta, \mathbf{x})$  denote the activation of that unit given the input  $\mathbf{x}$  and network parameters  $\theta$ . The objective for visualising what features this unit detects is then to alter the input  $\mathbf{x}$  to maximize the output in unit  $(i, j)$  with fixed network parameters, i.e.

$$\mathbf{x}^* = \underset{\mathbf{x} \text{ s.t. } \|\mathbf{x}\|=\rho}{\arg \max} h_{ij}(\theta, \mathbf{x}).$$

The authors state that the easiest way to do this optimization is to use *gradient ascent*, the same algorithm as *gradient descent* but moving in the opposite direction, this is due to that we have a maximization problem.

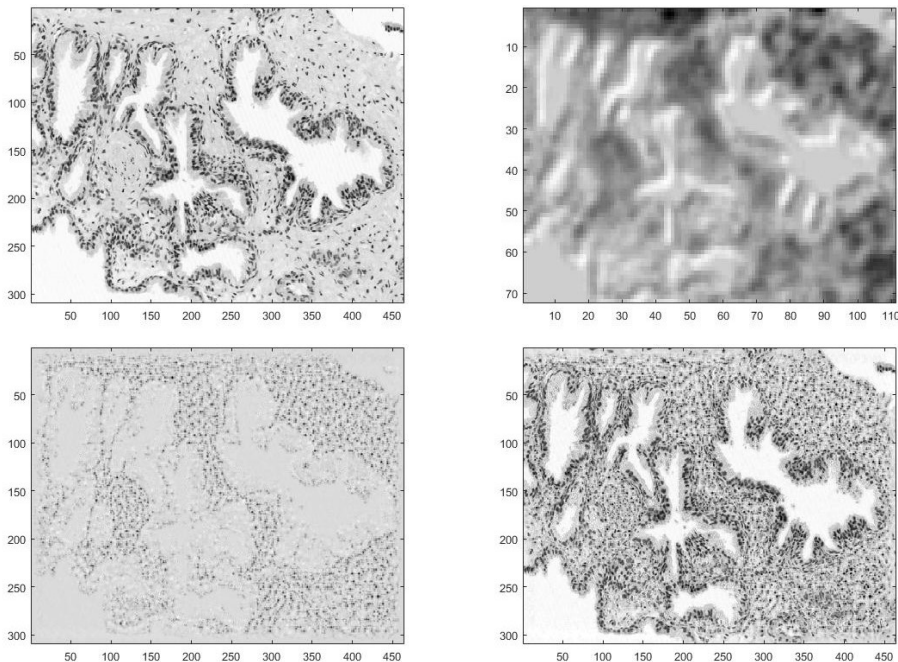


**Figure 6.1:** The first filters of an CNN and their impact on a benign input image.

The pragmatic way to do this is to use the back-propagation algorithm already implemented, which means that we use stochastic gradient ascent with momentum, as when training the network.

Furthermore we have done the simplification of only take one step in the ascending direction, since it is enough to see where and how  $h_{ij}(\theta, \mathbf{x})$  would be increased in the image. Also instead of tracking one neuron, the impact from a filter is found by setting the error to be back-propagated through other filters to be zero.

In figure 6.2 we see an example of a detected feature. We have chosen only to show the red channel because it makes the changes more vivid in this example. The first image is the original image. The second is the result after a filter chosen from the third convolution. We see that the image has been down sampled due to filtering and pooling. It seems at this stage that the edges of the glands gave high response. However, when back-tracking the gradients into the original image we may conclude that the changes for a larger response in this filter would be caused by an increased graininess in the stroma. The gradients were pictured in the third image and the added response in the last image.



**Figure 6.2:** Top left: The red-channel of a benign image. Top right: The output from a filter in the third convolutional layer. Bottom left: The back-propagated gradient from the chosen filter. Bottom right: The added image and gradient.

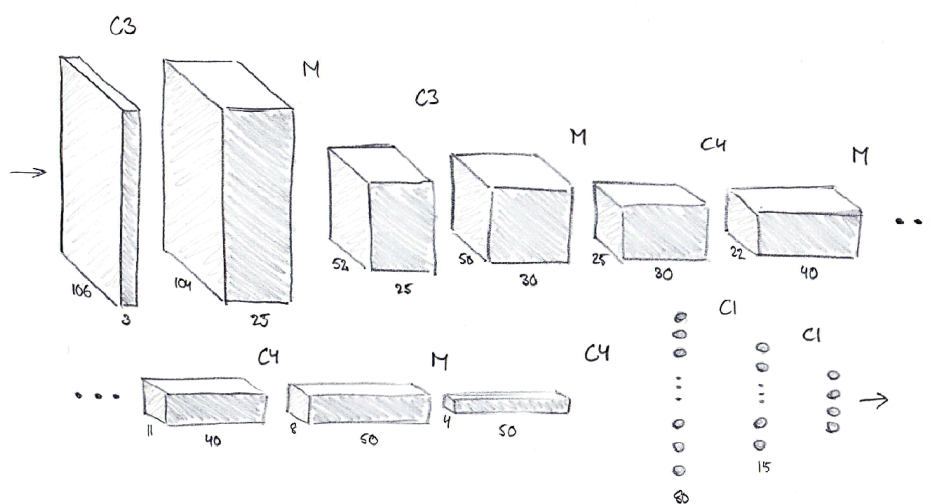




# 7

## Results

The network I have attained is pictured in figure 7.1. It consists of four blocks of alternating convolutional and max-pooling layers, and a two-layer MLP as conclusion. This design requires the training of 123885 weights.



**Figure 7.1:** Final network design. M is short for max-pooling and C means convolution, C3 means that 3x3 filters have been used in the layer. The numbers are the sizes of the images and how many there are in each layer. For example the input is a 106x106 image with 3 channels, after the first convolution we get smaller images (104x104) with 25 channels.

To derive this design, different patch-sizes, filter numbers, filter sizes and different normalisation techniques have been tested. An underlying goal has been to obtain a net big enough for the task but not much larger, as to decrease overfitting, reduce memory space used and increase training speed.

The result on the four cross-validation partitions can be seen in figures 7.2-7.3, the graphs to the right. The black lines are the training error, error on patches that are available for the algorithm to use. The blue lines are also per patch error, but for the validation set. These patches are extracted in the same way as the training patches, with rotation and

flipping. The images used in the validation set have not been available for the network to train on. Lastly the magenta line is the per image error, the result of entire images in the validation set. The images are cut up into patches by a sliding window and the total class label is decided on a vote between the patches. This is the most interesting result since the aim is to classify whole images correctly. The objective error, left graphs, takes the uncertainty of the per-patch result into account. If the output for a benign patch is  $(0.75, 0.2, 0.05, 0)$  then the error is 0, since the largest output is the one that correspond to benign tissue, but the objective error is 0.25 since there is still an uncertainty.

If the mean is calculated of all four runs we get the results seen in figure 7.4. Here the black line is the mean of per-image errors and the subject of interest. To get an estimate of the total per-image error, a mean of the last 25 epochs were made and an error rate of 7.3% was deduced. An epoch has passed when all data has been processed once and takes about 8.7 minutes in this case. This means that 100 epochs with 4-fold cross-validation takes 58 hours.

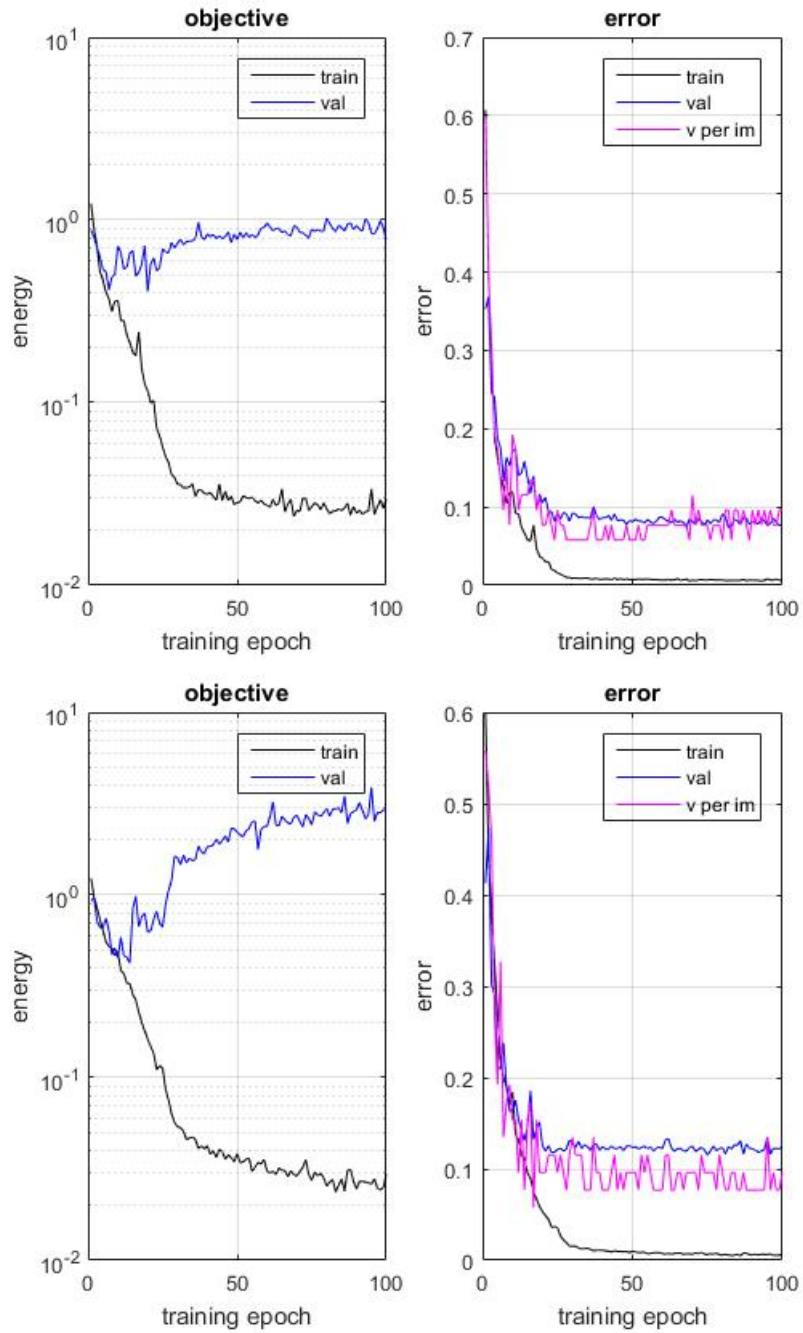
## 7.1 Misclassified images

A common way of investigating classification errors is too look at the confusion matrix. The rows represent the actual classes and the columns the predicted classes. The collective confusion matrix for all four runs after the last epoch looks like

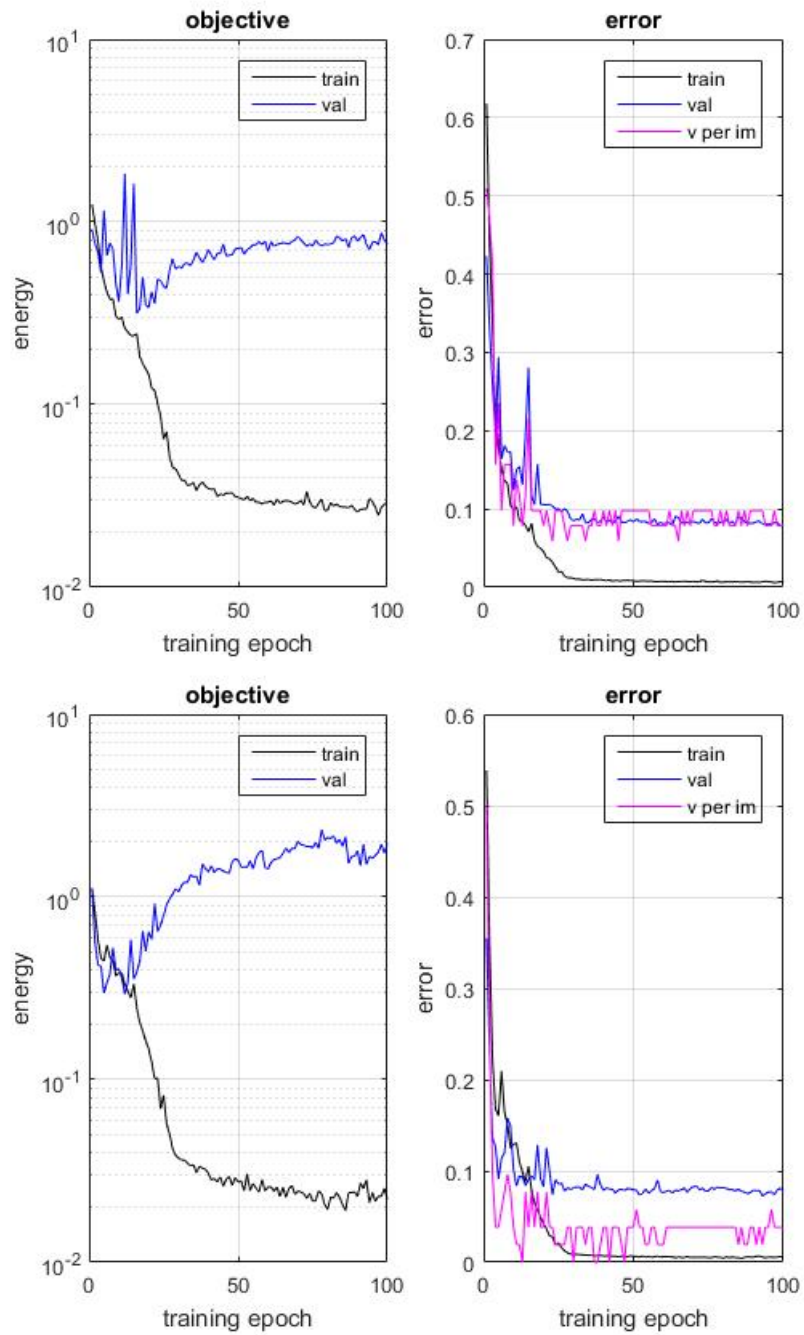
$$\begin{bmatrix} 51 & 0 & 0 & 0 \\ 3 & 46 & 3 & 1 \\ 0 & 6 & 43 & 0 \\ 0 & 3 & 0 & 52 \end{bmatrix}.$$

Thus all benign images were correctly classified, three Gleason 3 images were miss-classified as Gleason 4, three images as benign and one image was miss-classified as Gleason 5, and so on.

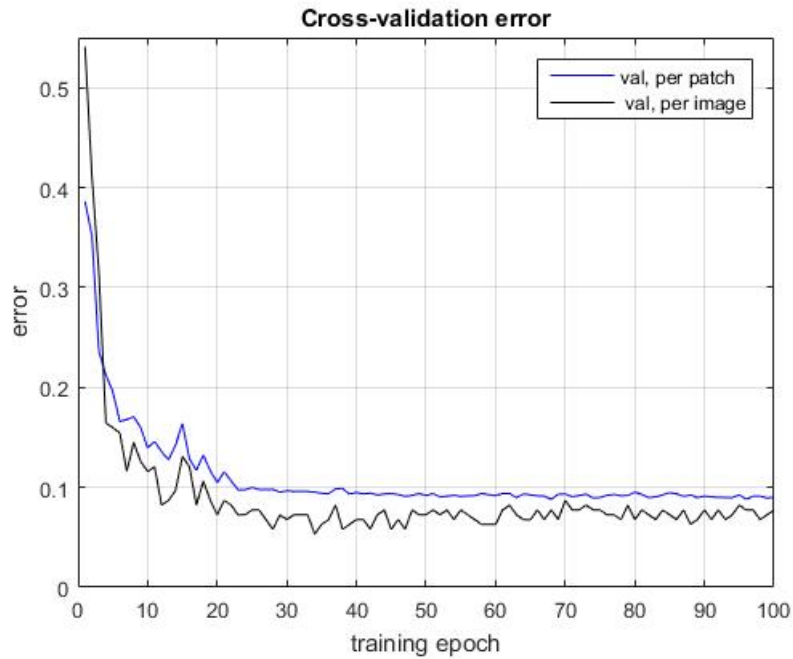
In figures 7.5-7.7 some of the misclassified images are pictured. The first image in each trio is the original image. The next image is the classification. Black means benign, red is Gleason 3, green means Gleason 4 and blue means that it has been classified as Gleason 5. The last is a combination of the two previous images.



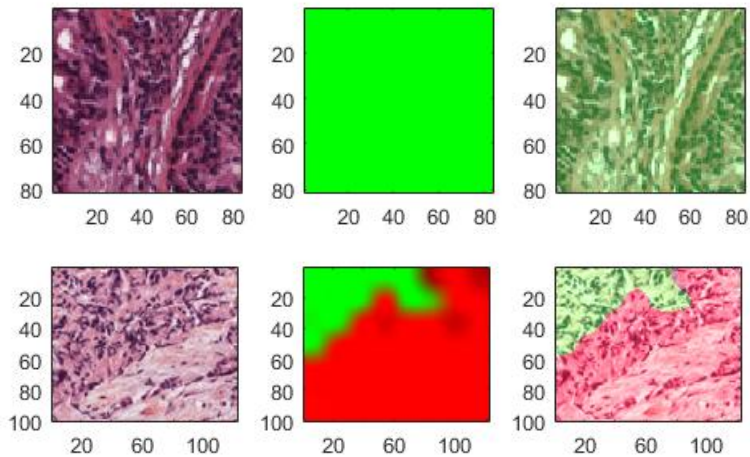
**Figure 7.2:** The results from the 4-fold cross-validation, first and second simulation.



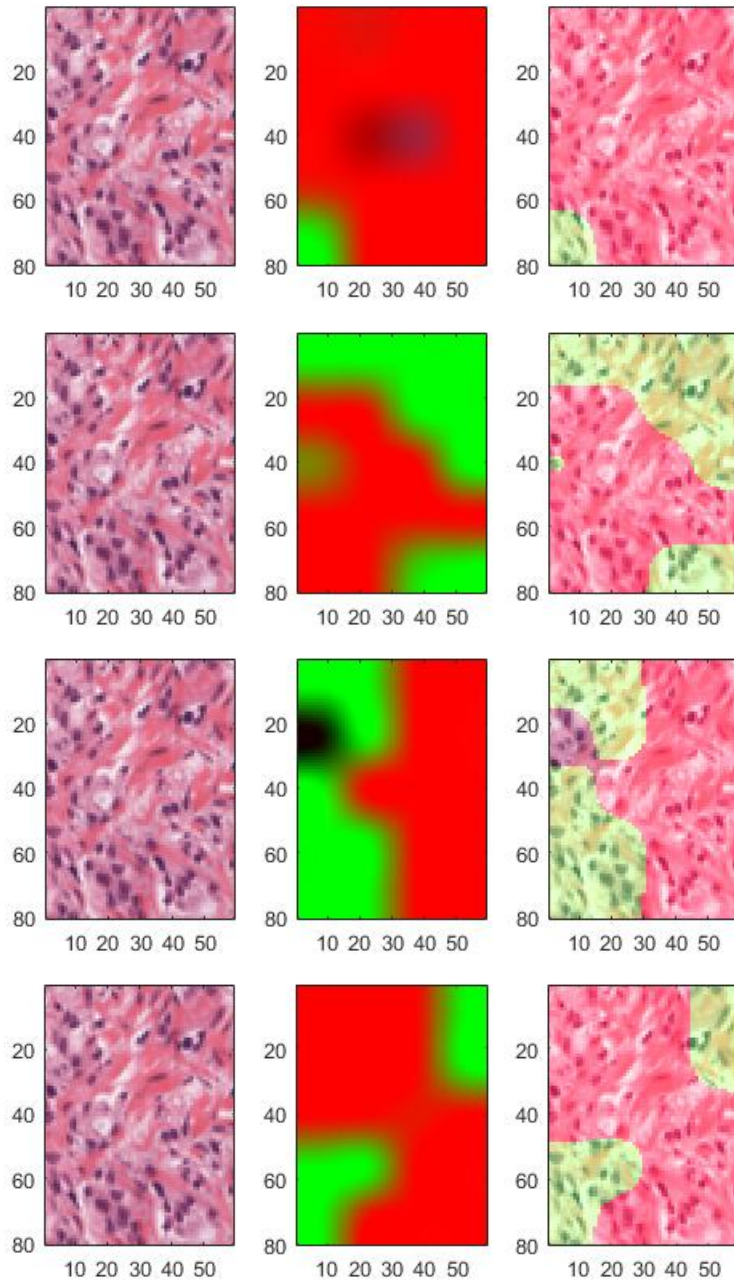
**Figure 7.3:** The results from the 4-fold cross-validation, third and fourth simulation.



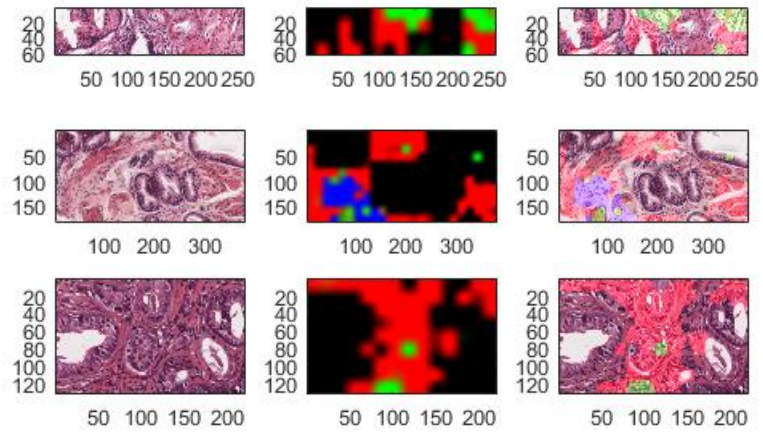
**Figure 7.4:** Mean result from four-fold cross-validation.



**Figure 7.5:** Misclassified images. The top images are classified as Gleason 4 but the label is Gleason 3, the bottom image set belong to are Gleason 5 but have been classified as Gleason 3.



**Figure 7.6:** Misclassified images. This is all the same image when the four nets have been applied, one for each cross-validation simulation. The first image is from when the image was in the validation set, otherwise the image has been part of the training set. The label is G4, which means that the classification ought to be green.



**Figure 7.7:** Misclassified images. These images are labeled Gleason 3, but are classified as benign tissue.





# 8

## Discussion

An error rate of 7.3% is not enough to be of clinical use. However, since different pathologists grade differently according to Persson et al. [3], a perfect score might not be achievable. The CNN in this thesis scores significantly better than the other attempts done on the same data set by [6] who scored 12.7% and [7] with a result of 10.2%. From this we may conclude that CNN is a method suitable for this problem.

The images in figure 7.7 are misclassified as benign when their label is Gleason 3. Some of the glands might be understandable to miss-classify as it has the classical papillary infolding usually seen in benign tissue. The difficulty of this classification problem is visible in figure 7.5 where the image sets look rather similar in structure and both are misclassified. The top image is supposed to be Gleason 3 with well-formed glands, so that the network find this image hard to classify is also understandable. An assurance of that the networks are not very overtrained can be seen in figure 7.6. The image contains little information and has been classified in the first net as Gleason 3 but the label is Gleason 4, the image was in the validation set for this network. What is of interest is that the image was not completely correctly classified when run through the networks that have used it for training. This difficulty shows that the networks do not "remember" the images pixel by pixel, which happens when the network gets more overtrained, but tries to find the patterns that are characteristic for different classes.

Now we have a quite coarse segmentation into classes. It would be easy to make it more specific by reducing the stride of the sliding window that extract patches for classification.

The result could be better. The network is still overtrained, which is visible through that the blue validation line is increasing again after a while in the objective error plot, see figures 7.2 and 7.3. We have however prevented serious overfitting since the training error never drops to zero. There are several other improvements that could be tested since many of the hyper-parameters have been locked, for example the filter size.

The best way of reducing error further might instead be to add more training data. 213 images are not much in a CNN environment. To be able to recognise a pattern the network has to have seen similar patterns in the training set, so a large variability in input means that a larger data set is required. Also odd biases might be avoided with a larger data set, as if there by chance happens to be more nerves depicted in one image class, then the network incidentally might take that as an indication of belonging to that class. More and unrelated data reduces this risk. In line with this reasoning all plausible different types of staining saturation should also be represented in the data set.

Another improvement would be to add more classes. An algorithm that assigns a class to the blank spots outside the specimen or to stroma that does not have a Gleason grading will look odd and probably reduce liability from an imaginary future user of this software. Other classes of interest might be colon tissue and the urethra. Given that the error rate is still low this would give the result a more professional look.

### 8.1 Future work

Apart from testing other net configurations, adding more training data and add more classes, a more segmentation oriented method might be interesting to use. Instead of classifying each patch and then assign the area covered of that patch to the class a finer edge of class boundaries might be found using a U-net described in [21]. Using feedback from several different scales in the process the network is able to segment on a much finer grid.

# Bibliography

- [1] Lindsey A Torre, Freddie Bray, Rebecca L Siegel, Jacques Ferlay, Joannie Lortet-Tieulent, and Ahmedin Jemal. Global cancer statistics, 2012. *CA: a cancer journal for clinicians*, 65(2):87–108, 2015.
- [2] Skriande brist på patologer skapar kris i cancervarden. <http://www.dn.se/debatt/skriande-brist-pa-patologer-skapar-kris-icancervarden/>, retrieved 2016-06-02.
- [3] Josefin Persson, Ulrica Wilderäng, Thomas Jiborn, Peter N Wiklund, Jan-Erik Damber, Jonas Hugosson, Gunnar Steineck, Eva Haglind, and Anders Bjartell. Interobserver variability in the pathological assessment of radical prostatectomy specimens: Findings of the laparoscopic prostatectomy robot open (lappro) study. *Scandinavian journal of urology*, 2014.
- [4] Lena Gorelick, Olga Veksler, Mena Gaed, José A Gómez, Madeleine Moussa, Glenn Bauman, Aaron Fenster, and Aaron D Ward. Prostate histopathology: Learning tissue component histograms for cancer detection and classification. *IEEE transactions on medical imaging*, 32(10):1804–1818, 2013.
- [5] Scott Doyle, Michael Feldman, John Tomaszewski, and Anant Madabhushi. A boosted bayesian multiresolution classifier for prostate cancer detection from digitized needle biopsies. *IEEE Transactions on Biomedical Engineering*, 59(5):1205–1218, 2012.
- [6] Giuseppe Lippolis. *Image analysis of prostate cancer tissue biomarkers*. PhD thesis, Lund University, Faculty of Medicine, 2015.
- [7] H. Källén, J. Molin, A. Heyden, C. Lundström, and K. Åström. Towards grading gleason score using generically trained deep convolutional neural networks. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 1163–1167, April 2016.
- [8] Geert Litjens, Clara I Sánchez, Nadya Timofeeva, Meyke Hermsen, Iris Nagtegaal, Iringo Kovacs, Christina Hulsbergen-van de Kaa, Peter Bult, Bram van Ginneken, and Jeroen van der Laak. Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis. *Scientific reports*, 6, 2016.
- [9] Simon Haykin. *Neural Networks and Learning Machines*. Pearson Education, 3 edition, 2009.
- [10] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Back-

- propagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.
- [11] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.
- [12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [13] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. Mnist handwritten digit database. <http://yann.lecun.com/exdb/mnist/>, retrieved 2016-05-26.
- [14] Vinay Kumar, Abul K. Abbas, Nelson Fausto, Stanley L. Robbins, and Ramzi S. Cotran. *Robbins and Cotran Pathologic Basis of Disease*. Elsevier Saunders, 7 edition, 2005.
- [15] Jonathan I Epstein, Michael J Zelefsky, Daniel D Sjoberg, Joel B Nelson, Lars Egevad, Cristina Magi-Galluzzi, Andrew J Vickers, Anil V Parwani, Victor E Reuter, Samson W Fine, et al. A contemporary prostate cancer grading system: a validated alternative to the gleason score. *European urology*, 69(3):428–435, 2016.
- [16] Oxford Visual Geometry Group. Matconvnet. <http://www.vlfeat.org/matconvnet/>, retrieved 2016-06-29.
- [17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [18] Ian Goodfellow Yoshua Bengio and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016.
- [19] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.
- [20] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341, 2009.
- [21] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

Master's Theses in Mathematical Sciences 2016:E40

ISSN 1404-6342

LUTFMA-3302-2016

Mathematics

Centre for Mathematical Sciences

Lund University

Box 118, SE-221 00 Lund, Sweden

<http://www.maths.lth.se/>