FACULTY OF SCIENCE AND TECHNOLOGY

MASTER THESIS

Study programme / specialisation:
Robotics and Signal Processing

The spring semester, 2022

Open / ~~Confidential~~

Author: Christopher Andreassen

Christopher Andreassen
(signature author)

Course coordinator: Professor Kjersti Engan

Supervisor(s): Professor Kjersti Engan, Professor Emiel Janssen, MD and PhD student Helga Hardardottir, PhD student Saul Fuster Navarro

Thesis title:
Melanoma prognosis prediction using image processing and machine learning

Credits (ECTS): 30

Keywords:
Melanoma, deep learning, supervised learning, convolutional neural network, prognosis

Pages: 70

+ appendix: 11 + 7z-file

Stavanger, 14th of June, 2022
date/year

Approved by the Dean 30 Sep 21
Faculty of Science and Technology

UNIVERSITY OF STAVANGER

Department of Electrical Engineering and Computer Science

Melanoma prognosis prediction using image processing and machine learning

Master's Thesis in Robotics and Signal Processing
by

Christopher Andreassen

Supervisor

Kjersti Engan, Professor

Co-supervisors

Helga Hardardottir, MD and PhD student

Emiel Janssen, Professor

Saul Fuster Navarro, PhD student

June 14, 2022

Abstract

Death of melanoma cancer is most common in Europe, and northern Europe has the second highest mortality rate of melanoma in the world, with 1.9 per 100 000 dying from melanoma in northern Europe in 2020.

Prognosis of melanoma is nowadays based on educated guesses by a pathologist, from analyzing patient tumors. Analyzing tumors takes much time, which limits the pathologist's capability for the number of tumors they are able to analyze in a given amount of time.

The primary objective of this thesis is to suggest a machine learning based method for predicting prognosis of melanoma, to aid the pathologist. The proposed method is based on the VGG16 architecture with pre-trained weights as the backbone, adding some fully connected layers. The network is trained and validated on whole slide images (WSI) from 51 patients with known melanoma prognosis, produced at Stavanger University Hospital. Regions of interest (ROI) areas in these images are marked by a pathologist.

A foreground segmentation algorithm for skin histological WSI is presented. Tiles are extracted from ROI areas in the WSIs, and resized to contain three different magnification levels, which are used in model training and validation. Multiple magnification levels are used to mimic the way a pathologist analyzes tissues at different magnifications.

Experiments are done by combining different magnification scales, utilizing tiles from respectively one, two and three magnification level(s) to train the models.

The best performing model used only one magnification scale at 20x. Cross validation results gave a F_1 score of 0.7667, and an area under the curve in a receiver operating characteristic curve of 0.81. This result is promising, considering the small number of patients in the dataset. For future work, the method has to be tested on a larger dataset. It is also recommended to test a larger set of possible hyperparameters and/or model architectures.

Acknowledgments

This thesis marks the end of my M.Sc. degree in Robotics and Signal Processing at the University of Stavanger.

I would like to express my deepest gratitude to my head-supervisor, Professor Kjersti Engan, for her invaluable patience, guidance and feedback through the entire work of this thesis.

I could not have undertaken this journey without co-supervisor, MD and PhD student Helga Hardardottir, who provided me with annotations and knowledge about the subject of melanoma. I am also grateful to my co-supervisor, Professor Emiel Janssen, for his presentation of melanoma and his vast knowledge in the field of pathology.

Thanks should also go to co-supervisor, PhD student Saul Fuster Navarro, for his guidance and invaluable feedback.

A final thanks to my friends and family, especially my parents and siblings, for encouraging me throughout five years of studies, leading to this thesis.

Contents

Glossary

Tis	Melanoma in situ
WSI	Whole slide image
H&E	Hematoxylin and eosin
GPU	Graphics processing unit
AI	Artificial intelligence
DNN	Deep neural network
CNN	Convolution neural network
ML	Machine learning
FC	Fully-connected
SGD	Stochastic gradient descent
ROC	Receiver operating characteristic
AUC	Area under the curve

1. Introduction

This chapter presents the motivations for this thesis, related work on predicting cancer prognosis and the thesis objective.

1.1 Motivation

In 2020, 120 771 people died from skin cancer, 57 043 of which being from melanoma skin cancer [1]. Of these melanoma deaths 26 360 people died in Europe, making Europe the continent with the most melanoma deaths in 2020. The second highest mortality rate in 2020 of melanoma in the world was found in northern Europe, with a mortality rate of 1.9/100 000 [2].

Melanoma is one of the most dangerous type of skin cancer in Norway, with a mortality rate of 6.8/3.8 per 100 000 (male/female) in 2020. The incidence rate has steadily increased each year from 1961 to 2020. This resulted in the incidence rate to increase 11.4/10.9% (male/female) from the five-year period 2011-2015, to the next one (2016-2020) [3].

Pathologists use much time analyzing cancer tumors, limiting their capability for the number of cancer tumors they can analyze in a given amount of time. This creates more work for the pathologists, which gives rise to faster methods to lower the time spent per tumor. One aspect of analyzing a cancer tumor is to find the prognosis of the cancer, which is used to decide the appropriate treatment for a patient. Prognosis relies on a pathologist's earlier experience and knowledge in the field, combined with statistical models based on prior patient histories [4]. This leads to an unsure prognosis that is only confirmed/disproved by the development of the tumor over time.

Modern technology has given rise to digitization of glass slides containing slices of a cancer tumor, resulting in whole slide images (WSI). Machine learning can use these WSIs to help find a likely outcome of a cancer tumor, by the use of prediction algorithms. This leads to reproducible and faster evaluation of a tumor, which can lead to better prognosis and less work-overload for the pathologist.

1.2 Related work

Machine learning (ML) has grown in popularity due to the increase of available data and process capabilities. This has lead to new and improved ML algorithms that applies to new tasks. ML is used to help medical doctors identify the prognosis of different type of cancers, e.g. breast cancer screening and cervical screening [5].

In the research part of this thesis, there were no published papers about prognosis prediction of melanoma using machine learning. Papers about prognosis prediction of other cancers is therefore mentioned here.

Lai et al. [6] proposed a method to predict 5-year survival of non-small cell lung cancer (NSCLC). A bimodal deep neural network (DNN) was made by merging the output layers of two DNNs, one feed with biomarkers and the other with clinical data. Zhang et al. [7] also proposed a method to predict 5-year survival of NSCLC. This was done using a feature transformation method based on convolution neural network (CNN).

Sun et al. [8] proposed a method called MDNNMD to predict prognosis of breast cancer. This was done by using multi-dimensional data, consisting of three datasets of medical data (gene expression, copy number alteration and clinical data). Each dataset was then train on separate DNN models, and the result was calculated as a weighted sum of the model's outputs. Prognosis prediction of breast cancer was done by Cheng et al. [9], by combining ensemble learning and a bimodal DNN.

1.3 Thesis objective

The main objective of this thesis is to propose a method to predict if a malignant melanoma in a patient will become metastatic within five years after diagnosis. Patients with metastatic melanoma within five years after diagnosis is classified as having bad prognosis, the rest is classified as having good prognosis. WSIs from malignant tissues with melanoma was annotated with good or bad prognosis by a pathologist. CNN was used to classify each tile in the WSIs as good or bad prognosis, using supervised learning. The result from this classification was then applied to predict each WSI to have good prognosis or bad prognosis.

2. Medical background

2.1 Overview of the skin tissue layers

The three layers of the skin are: epidermis, dermis and hypodermis, each of which is constructed differently and contains different parts of the skin's anatomy.

Epidermis

The most superficial layer of the skin is the epidermis, consisting of stratum basale (basal layer), stratum spinosum, stratum granulosum (granular layer) and stratum corneum (corneum layer), as shown in figure 2.1. The most superficial layer is here described as above the other layers. The basal layer contains stem cells that are constantly producing keratinocytes which makes keratin, and melanocytes which makes melanin, a brown pigment used to protect the cells in the skin from ultraviolet (UV) radiation. The keratinocytes are pushed from this layer thought the epidermis and die as it pass over to the outer layer of the epidermis (the corneum layer). Above the basal layer is straum spinosum, containing dendritic cells, one of the skin's first line of defense against incoming infections. Further up is the granular layer, where cells makes glycolipids, which glue cells in the epidermis together to hinder them from separating from each other. In the most superficial part of the skin is the corneum layer and consist of keratin and dead keratinocytes [10].

High-density areas of melanocytes are referred to as nevus and leads to a build up of melanin, leaving a brown color on the surface of the skin at this location. This nevus is often benign and harmless, but can sometimes be cancerous and therefore malignant.

Dermis and hypodermis

Dermis is placed underneath epidermis and consists of connective tissue containing extracellular components e.g. vessels, glands, hair follicles, fibroblasts and nerves. Hypodermis is located Under the dermis, deepest in the skin tissue, and consists primarily of fatty tissue [10][11].

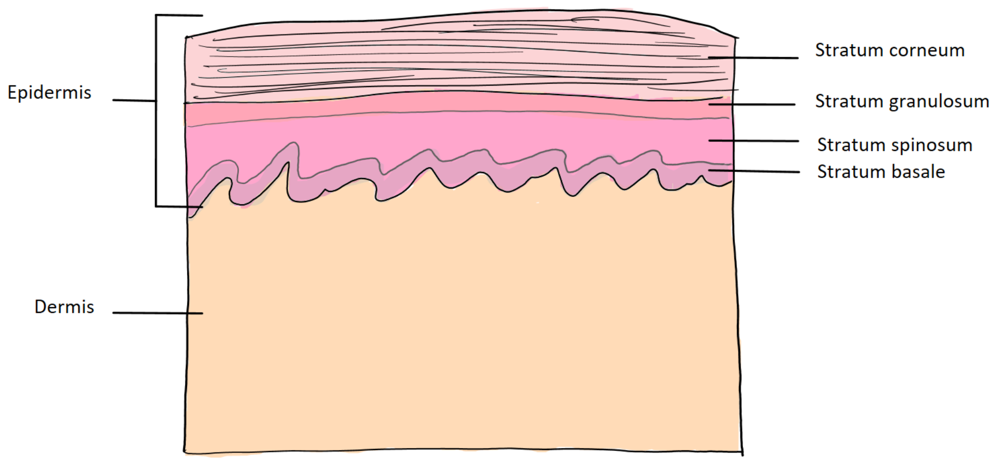


Figure 2.1: Sketch of epidermis and dermis.

2.2 Pathology and tissue processing

Pathology is the science of studying and diagnosing diseases through the examination of surgically removed organs, tissues, bodily fluids, or whole bodies during autopsies. Medical doctors in this field of study are called pathologists. They look at the anatomical structure and appearance of the cells in bodies and body tissues to find out if it is diseased. The area of studies for a pathologist is cellular adaptation to injury, necrosis, i.e. the death of body tissue, inflammation, wound healing and neoplasia (cells growing abnormally) [12]. Pathologists work with, among other diseases, melanoma skin cancer, where cellular patterns in a nevus are observed through a microscope to determine if the sample is benign or malignant.

For the cells in a nevus tissue to be visible through a microscope, the nevus has to first be sliced in to thin slices (typically 4-6 μm thick [13]) by a slicing tool. The most informative slices are then mounted on a glass slide and stained to enhance the visibility of the interesting areas in the tissue.

The stain hematoxylin and eosin (H&E) is widely used for examination of human tissue, due to its capability of highlighting the detailed structures of cells and tissues. H&E is a combination of two dyes: hematoxylin, with a blue color, and eosin, with a pink/red color. Hematoxylin binds to the nuclei, while eosin binds to cell membrane, proteins and the nucleolus [14]. Using this stain to examine a nevus makes the center of its cells appear in blue/purple, and areas around the center of the cells, muscle fiber, and more appear in pink/red, as shown in figure 2.2.

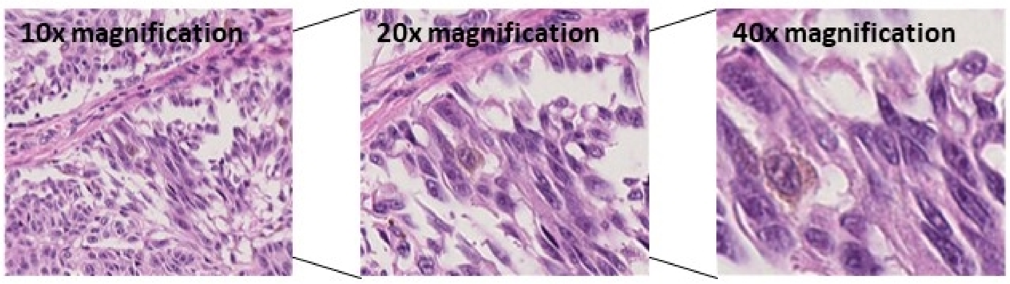


Figure 2.2: Tiles from part of a WSI containing malignant nevus.

2.3 Whole slide image

A whole slide scanner captures images of a tissue slide tile by tile or line by line, and assembles these images to generate a digital image, called whole slide image (WSI) [13]. This makes it possible to work with tissue slides on a computer. A WSI can be stored in different file-formats, depending on the scanner it's scanned with. The format used in this thesis is the ndpi-format, made by the Hamamatsu nanozoomer s60, which stores a WSI at different magnifications, making it easy to extract or look through different part of the image at these magnifications.

2.4 Melanoma

Melanoma is a cancer that occurs when the DNA in melanocytes, located in the basal layer, gets damaged and melanocytes begin to grow out of control. The emergence of melanoma can have different causes, e.g. through exposure to UV radiation from the sun [15].

2.4.1 Stages of melanoma

Melanoma becomes deadlier if it is not found in early stages and gets enough time to multiply and spread. It is therefore important to discover a melanoma tumor early. Medical doctors use stages to classify the spread of melanoma, which is described in a staging system. An example of such a system is TNM (primary **t**umor, lymph **n**ode, and **m**etastasis) system, which is based on three factors [16]:

- How far the primary **tumor** (**T**) has grown, and if it is ulcerated (any breakdown of the skin).
- If the cancer has spread to nearby lymph **nodes** (**N**).
- If the cancer has spread to a distant place in the body, from the primary tumor (**metastasis** (**M**)), such as distant lymph nodes or organs.

The five stages of the TNM system are summarized in figure 2.1, where stage 0 is the least invasive stage and stage 4 is the most invasive stage. Figure 2.3 shows the growth of the primary tumor in the different stages of the TNM system. A late diagnosis may result in a 5-year survival rate to decrease from 94.8/98.1% (male/female) to 36.6/55.1% (male/female), depending on the resulting stage [3].

2.4.2 Prognosis

The prognosis of a disease is a guesstimate about its outcome or course towards recovery. In this thesis, prognosis was defined by the presence/absent of cancer cells breaking away from where they first formed, and form new tumors in other parts of the body (metastasis) within five years of diagnosis. A patient with metastasis before five years after diagnosis had a bad prognosis, and a patient with metastasis five years after diagnosis had a good prognosis. Prognosis of melanoma is hard to estimate, because a tissue with good prognosis looks similar to a tissue with bad prognosis, as shown in figure 2.4, where the difference between the malignant lesions (2.4a and 2.4b) is small, compared to the benign lesion (2.4c).

Stage	Melanoma stage description
0	The cancer is only found in the epidermis. This stage is also called melanoma in situ (Tis).
1	The tumor is less than 2 mm thick and can be ulcerated (breakdown of skin over the melanoma), and it has not spread far in to the body or to the lymph nodes nearby.
2	The tumor is thicker than 1 mm (maybe thicker than 4 mm), can be ulcerated, and it has not spread far in to the body or to the lymph nodes nearby.
3	The tumor is more than 2 mm or 4 mm thick (Depending on the sub category), can be ulcerated, it has not spread far in to the body, but it has spread to nearby lymph nodes.[1]
4	The tumor can have any thickness, can be ulcerated, can have spread to nearby lymph nodes, and has spread to distant lymph nodes or organs (e.g. lungs, liver or brain).

[1] Stage 3 has four sub-categories (3A-3D), which is summarized here.

Table 2.1: Stages of melanoma according to the TNM system [16].

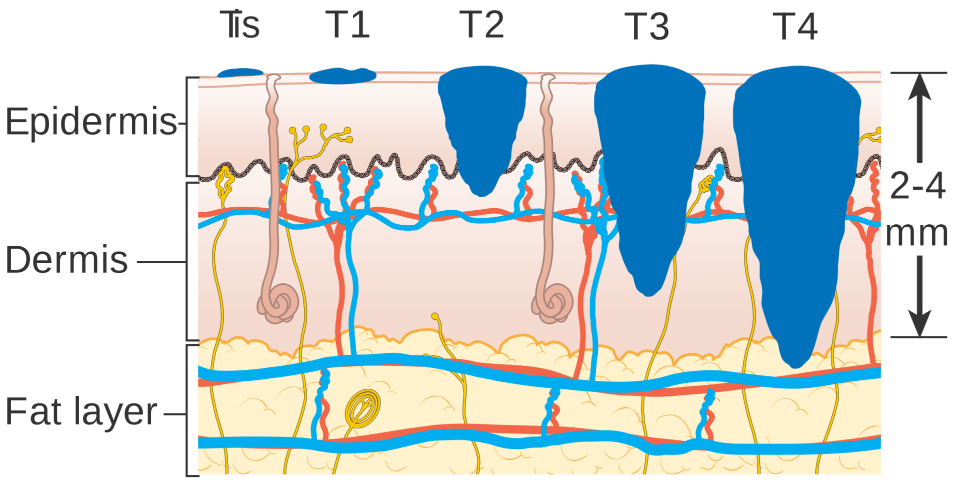
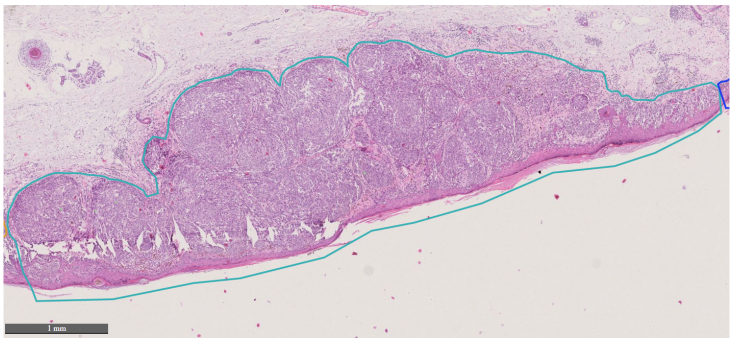
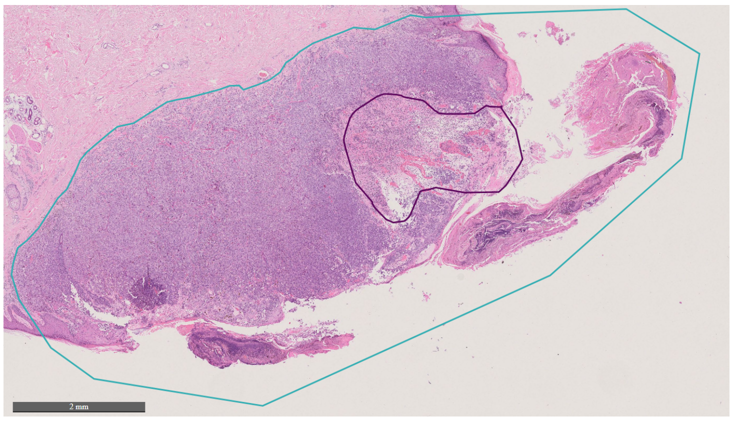


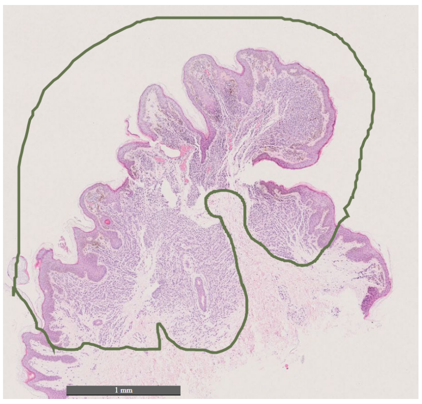
Figure 2.3: Illustration of the T stages in the TNM system.
The figure is reprinted in unaltered form from Wikimedia Commons, *File:Diagram showing the T stages of melanoma CRUK 373.svg*, licensed under CC BY-SA 4.0 [17].



(a) Malignant lesion, good prognosis.



(b) Malignant lesion, bad prognosis.



(c) Benign lesion.

Figure 2.4: Malignant lesion in tissue with good and bad prognosis, and benign lesion. Light blue surrounding contains malignant lesion, purple surrounding contains ulceration, and green surrounding contains benign lesion.

3. Technical background

3.1 Introduction to artificial intelligence

The goal of using artificial intelligence (AI) is to make intelligent machines/ computer programs, so that they can think and preform tasks meant for human brains. AI is used in multiple fields of study like cybernetics, information theory, statistics, computer science, among others.

3.1.1 Machine learning

Machine learning (ML) is a sub-field of AI and relies on data to learn independently, instead of following detailed instructions. A dataset is used to train a ML algorithm, and the resulting performance is evaluated against an independent test-set. The algorithm is learning if the performance of the test-set increase over time, as the algorithm is training. In short, the algorithm is learning the underlying features of the dataset. An important requirement for making a successful ML algorithm is large amount of data, so the algorithm has enough features to learn from a general representation of the data.

A machine learning algorithm is useful in many circumstances, e.g. in classification and clustering. Classification is used for labeling different categories of data, to for example separate images of different animals (dogs from cats), or predict the state of an illness in an medical image (good vs. bad prognosis of cancer). Clustering is used for grouping data in a data-set, to for example recognize behavior patterns (e.g. streaming service behavior, shopping behavior), or patterns in an image (group of objects).

The focus on ML has grown exponentially the past decade, with more than 32 times as many machine learning papers posted in 2018 as in 2009 [18]. This focus is still growing, with graphics processing units (GPU) computing power exponentially increasing every year. More and bigger datasets are created and challenges with focus on making the best preforming algorithm on a dataset is arranged, to be able to make better algorithms and to advance our understanding of machine learning.

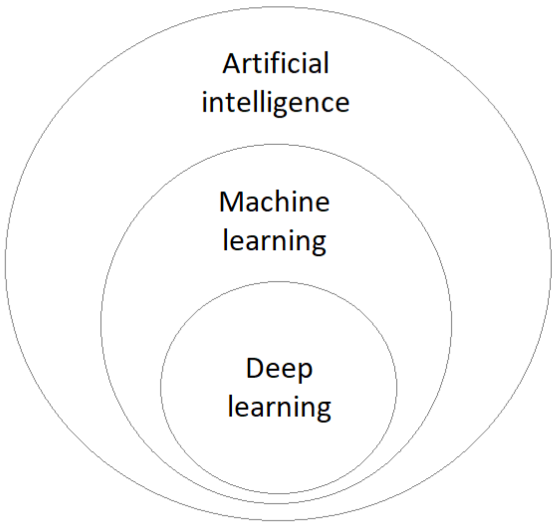


Figure 3.1: Relation between artificial intelligence, machine learning and deep learning

3.2 Deep learning

This section introduces neural networks and explains some deep learning networks and techniques.

3.2.1 Neural networks

Neural networks draws inspiration from the human brain, where an artificial neuron is a model of the biological neuron in the brain, as depicted in figure 3.2. This artificial neuron receives signals as inputs x_i, where $i = 0, ..., n$. These inputs are weighed against weights w_i and added together in a transfer function. The output of this function is equal to $\sum_{i=0}^{n} x_i w_i$. This output is passed through an activation function, which decides the value of the neuron's output and whether to activate the neuron or not. The weights can strengthen or weaken its corresponding input signal. Weight w_0 is often set to be a constant and input x_0 set to be equal to 1, in order to be able to shift the weighted sum with a constant value.

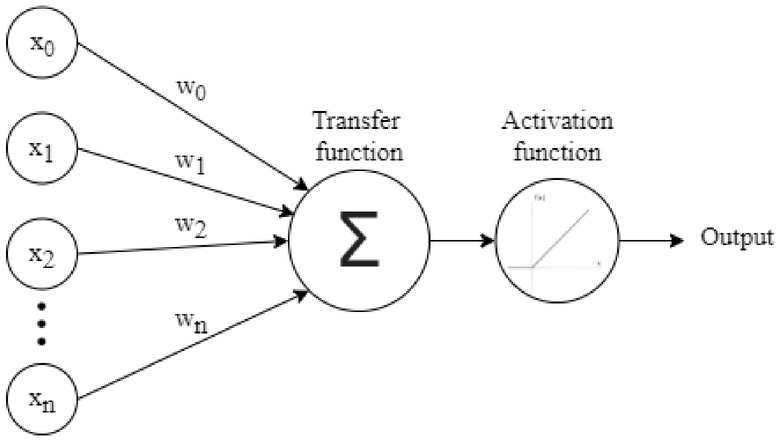


Figure 3.2: Artificial neuron, calculates a single output from a weighted sum of inputs.

A neural network contains a network of multiple artificial neurons, in the same way the brain contains a network of biological neurons. This neural network contains an input layer as its first layer, then one or more hidden layer(s) and an output layer as the last layer, as shown in figure 3.3. Each of these layers contains neurons, depicted as circles in the figure, and has each its neurons connected to all neurons in the next layer. These layers are called fully-connected layers (FC). The number of hidden layers in a neural network has grown the last decade, following the demand for more complicated networks, able to solve more complicated tasks than before. This has lead to deeper networks, with many hidden layers, called deep neural networks (DNN).

3.2.2 Convolutional neural networks

A convolutional neural network (CNN) is a neural network where some of the layers contain convolution layers. These layers use convolution operations between its input and an kernel with weights, to detect features from the input. The kernel weights are updated while the CNN is training.

An example of a convolution operation is shown in figure 3.4. The image in figure 3.4b can be seen as an input to a convolution layer. The kernel in figure 3.4a is a 3 by 3 sized kernel that amplifies main-diagonal edges (the diagonal going from top-left to bottom-right). The convolution between the

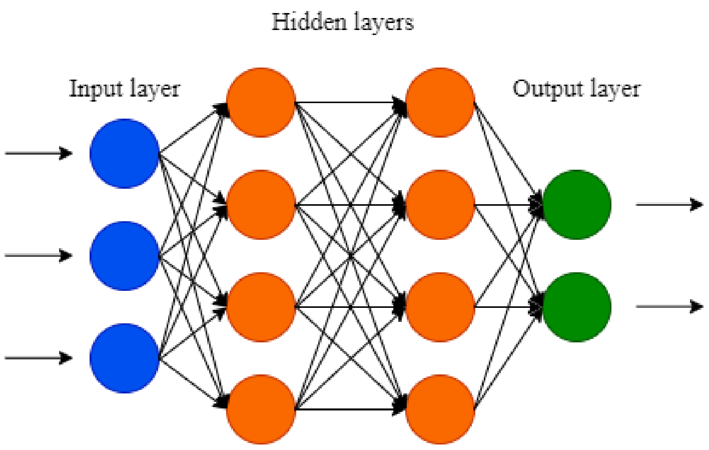
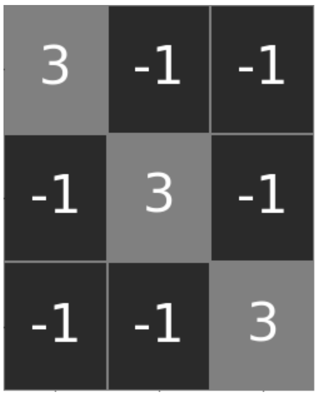


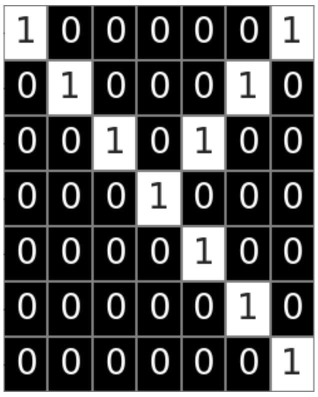
Figure 3.3: Example of a neural network with one input layer, two hidden layers and one output layer.

image and the kernel results in an amplification along the main-diagonal of the image, and a dampening along the anti-diagonal (the diagonal going from bottom-left to top-right) of the image, as shown in figure 3.4c. This kernel can therefore be seen as an filter used to find features similar to the kernel. A convolution layer can have multiple filters like this, able to find multiple features from one input. The convolution layer therefore makes CNNs useful for finding features in images.



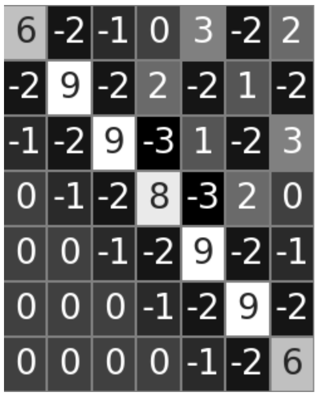
3	-1	-1
-1	3	-1
-1	-1	3

(a) Kernel K



1	0	0	0	0	0	1
0	1	0	0	0	1	0
0	0	1	0	1	0	0
0	0	0	1	0	0	0
0	0	0	0	1	0	0
0	0	0	0	0	1	0
0	0	0	0	0	0	1

(b) Image X



6	-2	-1	0	3	-2	2
-2	9	-2	2	-2	1	-2
-1	-2	9	-3	1	-2	3
0	-1	-2	8	-3	2	0
0	0	-1	-2	9	-2	-1
0	0	0	-1	-2	9	-2
0	0	0	0	-1	-2	6

(c) X∗K

Figure 3.4: Convolution between an image and a kernel.

The first layers of a CNN can be seen as a feature extractor, containing convolution layers that finds features of the input. The last layers of a CNN can be seen as a classifier, which often consists of fully-connected layers, that decides upon an outcome from the feature data made by the feature extraction layers.

Pooling layer

A pooling layer down-samples its input to make the network more compact and to reduce the computational burden. The most common types of pooling layer are max-pooling and average-pooling. Max-pooling finds the maximum value in an area of the input, and replaces this area with its maximum value. This operation can be done on the entire input, by using a set sized kernel and slide it a number of strides along the entire input. Max-pooling is shown in figure 3.5, where a 2 by 2 sized kernel strides 2 pixels every time the max-operation is used. Average-pooling is similar, but uses average instead of max.

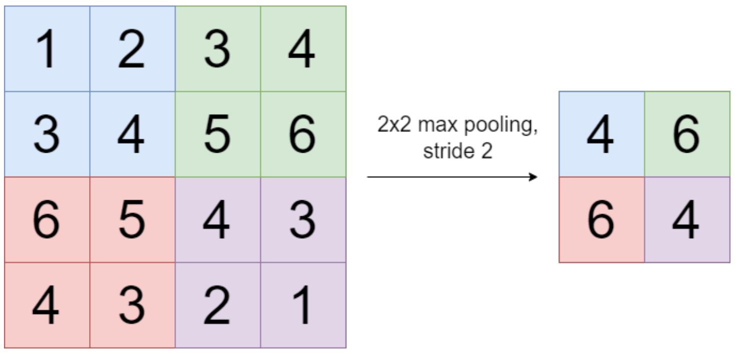


Figure 3.5: Example of max-pooling on a single input matrix.

Flatten layer

A flatten layer is used to transform a multidimensional input into a one dimensional vector. This is called flattening and is shown in figure 3.6. The flatten layer is often used to transform a matrix from a pooling layer into a vector, and pass this vector through a fully-connected layer to be further processed.

Dropout layer

Networks with small amounts of input data, from a dataset, may suffer from overfitting, where the network has learned too many features from too little data. This causes the network to not be able to correctly represent the entire dataset. Overfitting will lead to some features being prioritized higher in the

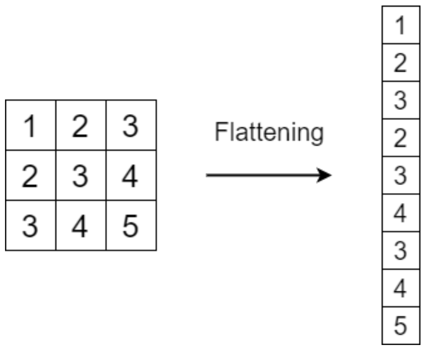


Figure 3.6: Example of flattening on a single input matrix.

network than they are in general, which can lead to a poor performance of the network. Dropout layers is used to solve this problem. This layer sets the input of a given percentage of neurons, in a fully-connected layer, to zero. The result of this is an simpler FC layer with deactivated neurons, forcing the network to be less dependent on particular neurons.

3.2.3 VGG16 - a convolutional neural network

VGG16 is a CNN architecture proposed by K. Simonyan and A. Zisserman in 2014 [19]. VGG16 took first and second place in their submission for the ImageNet Challenge in the same year. The architecture for this network is shown in figure 3.7, with 16 trainable layers. Weights of this network, trained on ImageNet, are located in most framework for deep learning (e.g. PyTorch and TensorFlow), which have made VGG16 a popular network to use.

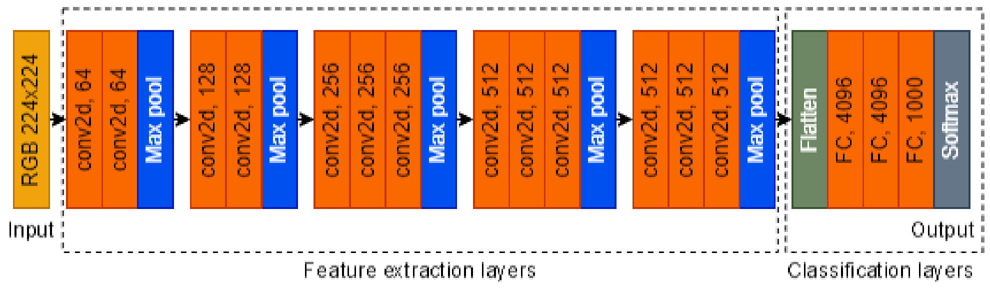


Figure 3.7: VGG16 architecture. Convolutional layers with said number of 3x3 filter kernels and ReLu activation function. Fully-connected (FC) layers with said amount of neurons. The architecture is described in [19].

3.2.4 Transfer learning

A pre-trained neural network is a network that is trained on another dataset. This dataset can have a wide amount of data, with a lot of basic features that can be found in many other datasets. Having knowledge about these features when training a new network can therefore be useful. Transfer learning is a machine learning technique, where a pre-trained network is re-purposed to be used in another task. The layers in the pre-trained network can be used in its entirety, or partially switched out with other layers, which have to be trained from scratch.

A layer in a pre-trained network can be set to not be trained. A non-trainable layer is often referred to as a frozen layer, and can be useful for not altering weights in a layer that is important for finding basic features. The first layers in a pre-trained network is often useful to freeze, because it is mostly these layers that learn basic features [20].

3.3 Supervised learning

Supervised learning is the machine learning task of learning the relationship between a list of inputs and a corresponding list of outputs. The inputs are normally vectors or matrices and the outputs are normally single values. For every output y in the dataset there has to be a relationship to its input x in the dataset, such that $y = f(x)$. The goal of using supervised learning is to train a network to approximate this relationship and try to find $y = g(x)$, where $g(x)$ is the result of input x going through the network.

3.3.1 Train and validation

A dataset can be divided in to train set and validation set. The training set is used to fit parameters of a network during training. The validation set is used to check how well the network is training, by testing the network on new data, while training. This set has to have no influence on the network, in order for the test to be a valid estimate on how well the network is training. A test set can also be introduced, to test the network after training, but it's often desirable to leave this data in the train set and validation set when working with a small dataset. The reason for this is to strengthen the training of the network.

3.3.2 Cross-validation

Cross-validation uses different parts of a dataset to train and validate a network on different iterations. This is done to see if the performance of the network changes depending of which part of the dataset that is used for validation.

Stratified K-fold

K-fold divides the dataset in to K groups, called folds, where one fold of the dataset is used for validation and the rest of the dataset is used for training. This is done K iterations to make K datasets of training- and validation-sets, with each having a unique validation-set. Stratified K-fold adds one more requirement to the setup of the dataset, where the same number of data from each class has to be represented in each fold, as seen in figure 3.8. This results in a good representation of each class in each iteration. K-fold can be used in cross-validation to acquire a unique validation set in each iteration.

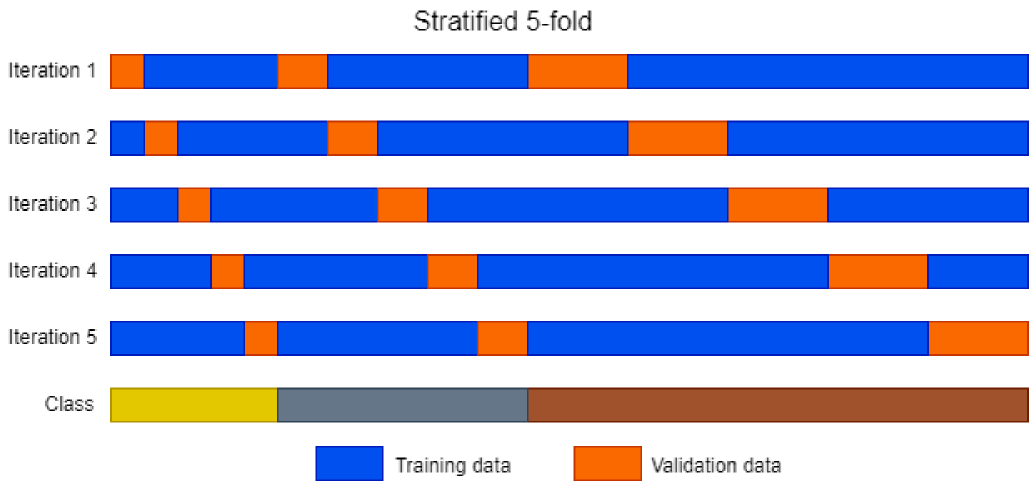


Figure 3.8: Example of stratified 5-fold validation. The dataset is split in to training data and validation data for each iteration, depending on the class distribution.

3.3.3 Gradient descent

Gradient descent is an algorithm used to find a local minimum of a differentiable function. A non-zero valued gradient at point a in a function, shows the direction where the function increases most quickly from the point. This means that the quickest way from point a to a local minimum is the opposite direction of the gradient at point a, $-\nabla f(a)$. Equation 3.1 shows the formula for gradient descent, where a point closer to a local minimum is defined to be a set of steps along a descending direction from an original point. These amount of steps is defined using a variable called learning rate, μ.

$$a_{i+1} = a_i - \mu \nabla f(a_i) \tag{3.1}$$

Figure 3.9 shows an example of gradient descent in an tree-dimensional space. For a point to move to the function's minimum value the learning rate has to be grate enough to get out of other minimums, but small enough for the point to not go past and ignore the function's minimum value.

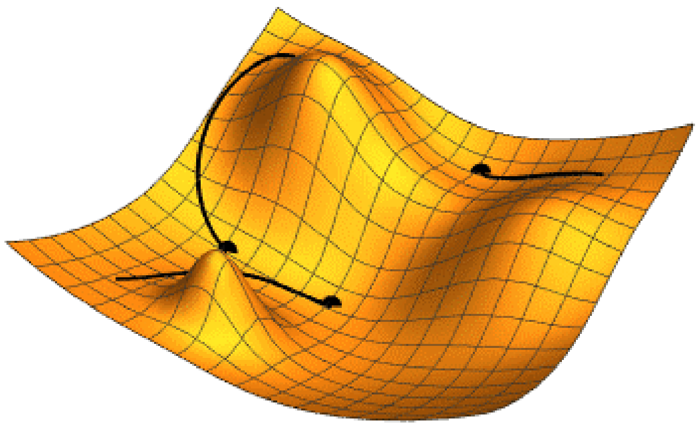


Figure 3.9: Example of gradient descent with tree different initial conditions.
The figure is part of a GIF from Wikimedia Commons, *File:Gradient descent.gif*, in public domain under CC0 1.0 [21].

In neural networks, a loss function is the difference between an outcome produced by a network and the expected outcome. Gradient descent is used to minimize this loss function in a neural network.

3.3.4 Early stopping

A network is trained with a whole train set for a set amount of cycles, called epochs. If this network is trained on the training data for too many epochs it tends to be too well fitted to this data. This leads to a network that is good at finding features from the training, but bad at finding features for data in general. The network is therefore overfitted to the training data. Early stopping is used to hinder this from happening, by stopping the training before it is overfitted. This is done by monitoring the validation during training, if the validation does not improve after a predetermined number of epochs, the training will be stopped. The weights from the best performing epoch are then stored.

3.3.5 Evaluation metrics

A neural network has many hyperparameters that needs to be defined before the network can be trained, e.g. learning rate, number of neurons in each layer, dropout rate, and batch size, the number of data processed before the network updates. There are usually no obvious ways of determine the optimal hyperparameters, which is way one typical train a network starting with a set of hyperparameters. These parameters are then adjusted depending on how the network performs.

Neural networks can be evaluated using evaluation matrices. These matrices use a validation set to gather statistical data from the training of the network. This is then used to find a network's performance.

Confusion matrix

A confusion matrix shows a summary of the performance of an algorithm on different classes, which is useful in classification tasks. Each row of the matrix shows the instances of actual conditions for a class, and each column shows the instances of predicted condition for each class, as shown in figure 3.10. The classes in a confusion matrix represents a presence or absence of a condition (e.g. melanoma / benign, positive / negative prediction), defined by the task. Class 1 is here the class with a condition and class 2 is the class without the condition. A list describing parameters in a confusion matrix is shown in table 3.1.

Accuracy

Accuracy describes how much data in a dataset is correctly predicted. This is calculated as shown in equation 3.2, where we want the result of the equation to be as close as possible to 1. Accuracy can be a good indication of how well a neural network is performing in a balanced dataset.

$$ACC = \frac{TP + TN}{P + N} \tag{3.2}$$

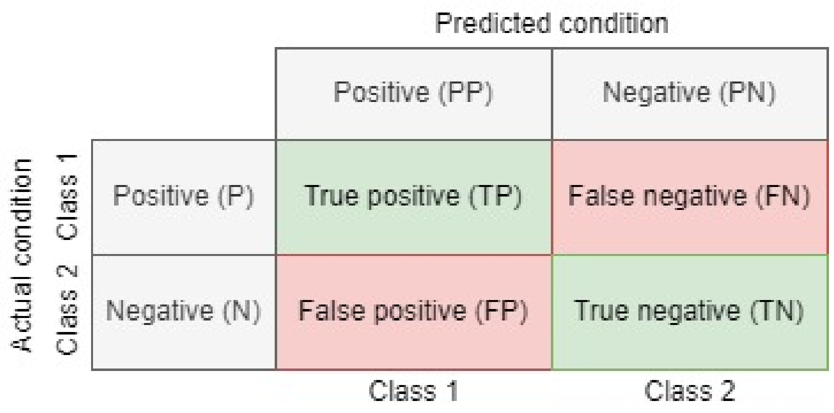


Figure 3.10: A confusion matrix with two classes.

Sensitivity

Sensitivity is the amount of actual positive instances that are correctly predicted. It is desirable to have the number of true positive (TP) instances as close as possible to the actual number of positive (P) instances. Therefore, looking at the equation for sensitivity (equation 3.3), one can see the sensitivity should be close to one for a well trained network. This value is useful for noticing how many instances that have a condition, are predicted to have the condition.

$$SEN = \frac{TP}{P} \tag{3.3}$$

Precision

Precision, also called positive predictive value (PPV), is the amount of positive predicted instances that are correctly predicted. It is desirable to have the number of true positive (TP) instances as close as possible to the number of instances predicted to be positive (PP). Therefore, looking at the equation for precision (equation 3.4), one can see the precision should be close to one for a well trained network. This value is useful for noticing how many instances are predicted to have a condition, have the condition.

$$PPV = \frac{TP}{PP} \tag{3.4}$$

Parameters	Description
True positive (TP)	The number of instances correctly indicating that a condition is present.
False positive (FP)	The number of instances wrongly indicating that a condition is present.
True negative (TN)	The number of instances correctly indicating that a condition is absent.
False negative (FN)	The number of instances wrongly indicating that a condition is absent.
Actual positive (P)	The number of actual positive instances.
Actual negative (N)	The number of actual negative instances.
Predict positive (PP)	The number of instances predicted to be positive.
Predict negative (PN)	The number of instances predicted to be negative.

Table 3.1: Description of parameters used in confusion matrices.

Specificity

Specificity is the amount of actual negative instances that are correctly predicted. It is desirable to have the number of true negative (TN) instances as close as possible to the actual number of negative (N) instances. Therefore, looking at the equation for specificity (equation 3.5), one can see the specificity should be close to one for a well trained network. This value is useful for noticing how many instances without a condition are predicted to not have the condition.

$$SPC = \frac{TN}{N} \tag{3.5}$$

$\mathbf{F_1}$ score

F_1 score is used for combining sensitivity and precision in to an single metric by using their harmonic mean, as shown in the equation for F_1 score in equation 3.6. The result of this metric is then a measurement of a system's accuracy, by looking at the amount of true positive in relation to false negative and false positive. This measurement can be a good final score for evaluating a system.

$$F_1 = \frac{2PPV \times SEN}{PPV + SEN} = \frac{2TP}{2TP + FP + FN} \tag{3.6}$$

4. Material and previous work

4.1 Data material

The datasets in this thesis consist of tiles from 51 WSIs from 51 patients, annotated by co-supervisor H. Hardardottir. Of these WSIs, 25 patients has good prognosis of malignant melanoma and 26 has bad prognosis. The prognosis depends on metastasis or no metastasis withing five year, which is known in these WSIs. The WSIs is stained with H&E stain and scanned at 40x magnification, with a Hamamatsu nanozoomer s60 scanner. An overview of the WSIs is shown in figure A.1.

A variable D referrers to a dataset in the form of start coordinates. The datasets contain a number of start coordinates of tiles from one or multiple magnifications, assigned with variable mj, extracted from a magnification level, assigned with variable i. Stratified 5-fold is used to divide the data in to five iterations of training data and validation data. Training datasets are assigned with variable t and validation datasets are assigned with variable v, both followed by a variable k, indicating the iteration the dataset is from. The dataset variable format is: $D^{i,mj}_{t/vk}$, and the variable options are shown in table 4.1. A detailed overview of the dataset used in this thesis is shown in table 4.2. Mono-, di-, and tri-scale is used to refer to the number of magnification levels used for every data input. Mono-scale refer to one magnification level, di-scale refer to two and tri-scale refer to three.

Variable	Values	Description
i	20, 40	Magnification level where start coordinates of tiles are extracted from. i refer to ix magnification.
mj	$m10$, $m20$, $m40$, mDi, $mTri$	Magnification level(s) of extracted start coordinates. j being a number refer to jx magnification, mDi refer to 20x and 40x magnification and $mTri$ refer to 10x, 20x and 40x magnification.
t/v	t, v	t being training dataset and v being validation dataset.
k	Integer in range 1 to 5	Iteration the dataset is from.

Table 4.1: Variables used to explain a dataset $D_{t/vk}^{i,mj}$.

4.2 Previous work

This section presents work done by co-supervisor H. Hardardottir, R. Wetteland [22] and N. Inkawhich [23].

4.2.1 Annotations

All WSIs were annotated by co-supervisor H. Hardardottir, a pathologist at Stavanger University Hospital. Examples of these annotations are shown in figure 2.4. A program for annotating WSIs, made by students at University of Stavanger, was used to annotate the WSIs in this thesis. This program stores coordinates and other information (e.g., tag and creator) about the annotation in XML files, with the setup shown in listing 4.1. Annotation coordinates from a WSI was used to make an image mask containing the annotated area, as described in section 5.1.

Listing 4.1: XML file with annotation data. Each XML file belongs to a WSI and contains coordinates constituting annotated surrounding around an area in the WSI.

```
<Annotations>
 <Annotation>
  <Regions>
   <Region tags="Lesion malign" name="" creator="helgah" grade="1">
    <Vertices>
     <Vertex X="22275.354502193073" Y="76000.51584277466" Z="0"> </Vertex>
     <Vertex X="22262.314484580576" Y="76013.55586038716" Z="0"> </Vertex>
                                    ...
     <Vertex X="22275.354502193073" Y="76000.51584277466" Z="0"> </Vertex>
    </Vertices>
   </Region>
```

4.2.2 Tile extraction

A tile extraction algorithm made by R. Wetteland [22] is used in this thesis under the GNU General Public License v3.0. The pipeline of this algorithm is displayed in figure 4.1, conveying how the algorithm extracts x and y coordinates from one magnification, representing tiles in a WSI. An image mask is used to extract as many valid start coordinates of tiles as possible, containing a predefined percentage of the mask. A predefined magnification level is chosen as a reference for tile extraction. From this magnification level, coordinates of tiles are extracted and resized to obtain coordinates from multiple magnifications. A visual representation of the coordinate extraction is shown in figure 4.2, where start coordinates from three tiles (top of figure) with magnifications 10x, 20x and 40x are extracted from a WSI. Extracted coordinates are saved as a pickle file.

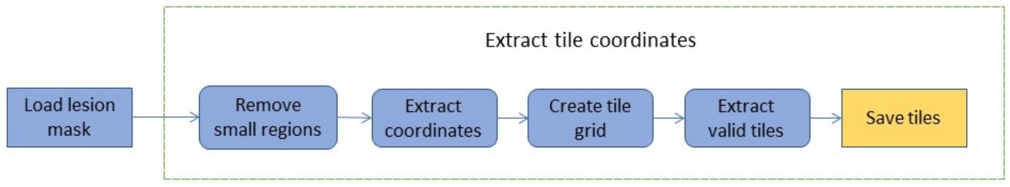


Figure 4.1: Algorithm for extracting tile coordinates, made by Rune Wetteland [22].

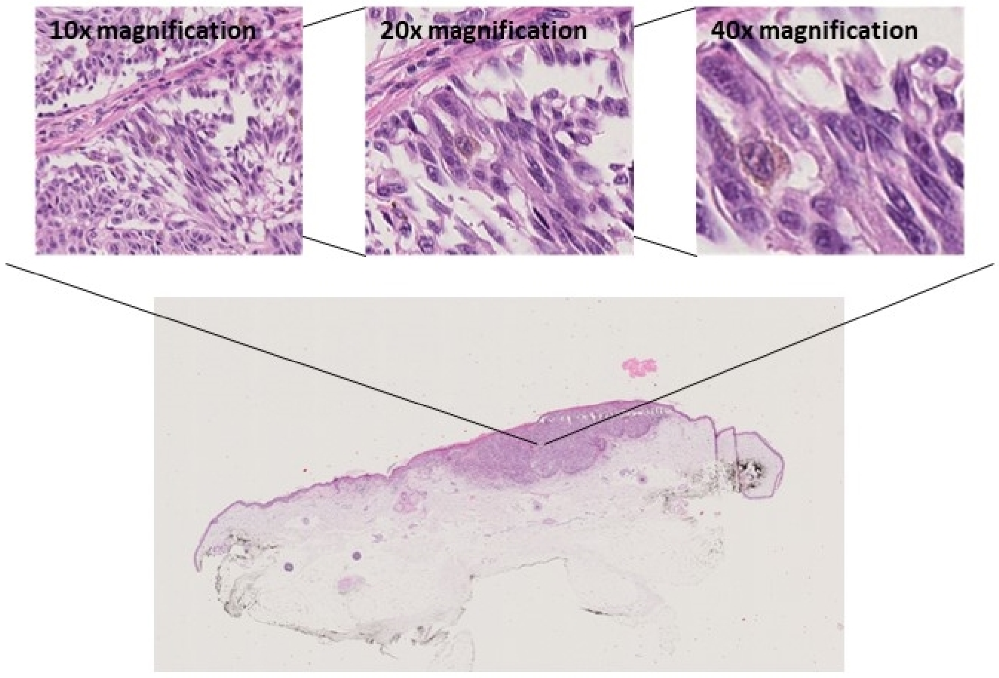


Figure 4.2: Extracted tiles from part of a WSI containing malignant nevus.

4.2.3 Network

The code for making and training the network is based on code made by N. Inkawhich [23], found on pytorch.org and used under the BSD license. This code does the following:

- Initializes a pre-trained model and freezes layers.
- Setup for data augmentation and normalization.
- Creates optimizer that updates desired parameters.
- Trains the network and makes training and validation history of the training.

This code is described in section 5.2, including changes listed in table 4.3.

Bad prognosis Good prognosis				
Dataset	Tiles	% of all tiles	WSIs	Avg. tiles per WSI
$D_{t1}^{20,mj}$	4 876 4 292	41.0% 36.1%	20 20	244 215
$D_{v1}^{20,mj}$	1 482 1 250	12.5% 10.5%	6 5	247 250
$D_{t2}^{20,mj}$	5 108 4 505	42.9% 37.9%	21 20	243 225
$D_{v2}^{20,mj}$	1 250 1 037	10.5% 8.7%	5 5	250 207
$D_{t3}^{20,mj}$	5 108 4 471	42.9% 37.6%	21 20	243 224
$D_{v3}^{20,mj}$	1 250 1 071	10.5% 9.0%	5 5	250 214
$D_{t4}^{20,mj}$	5 108 4 463	42.9% 37.5%	21 20	243 223
$D_{v4}^{20,mj}$	1 250 1 079	10.5% 9.1%	5 5	250 216
$D_{t5}^{20,mj}$	5 232 4 437	44.0% 37.3%	21 20	249 222
$D_{v5}^{20,mj}$	1 126 1 105	9.5% 9.3%	5 5	225 221
Total	6 358 5 542	53.4% 46.6%	26 25	245 222
	11 900	100%	51	233
$D_{t1}^{40,mj}$	5 000 4 911	39.5% 38.8%	20 20	250 246
$D_{v1}^{40,mj}$	1 500 1 250	11.8% 9.9%	6 5	250 250
Total	6 500 6 161	51.3% 48.7%	26 25	250 246
	12 661	100%	51	248

Table 4.2: Overview of datasets, top row for each dataset is extracted from bad prognosis WSI and the bottom from good prognosis. *Tiles* shows number of tiles corresponding to start coordinates from one magnification level, and *WSIs* shows number of WSIs the tiles are extracted from.

Change	Description
VGG16	VGG16 network is used, with pre-trained weights.
Early stopping	Added early stopping to training function ($train\_model()$), mentioned in section 5.2.
Cross validation	Cross validation is implemented, as described in section 5.2.
Make data sets	Method for making training and validation datasets was made, as described in section 5.2.1.
Classification layer	Froze feature extraction layer weights and changed number of output neurons. This is described in section 5.2.2.
Save data	Information about the run, best model(s) and prediction data from validation is saved.

Table 4.3: Changes to the code from [23].

5. Methods

This chapter explains methods used to process, extract and classify melanoma lesions in WSIs containing malignant nevus, utilizing traditional image processing and deep neural networks.

The proposed method is shown in figure 5.1 and consists of a preprocessing part and a training part. The preprocessing part extracts start coordinates of tiles from lesion in WSIs, and the training part makes training and validation datasets of these coordinates, and trains a CNN using these datasets.

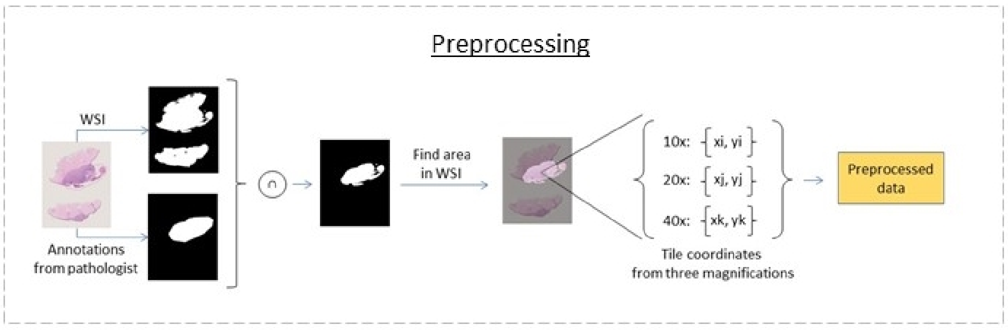
5.1 Preprocessing

Extraction method made by R. Wetteland [22] was used to extract start coordinates of tiles in a WSI, by the use of an image mask, as displayed in figure 5.1. This image mask was made from the intersection between a mask containing tissue in WSI (tissue mask) and mask containing annotated lesion in WSI (annotated mask).

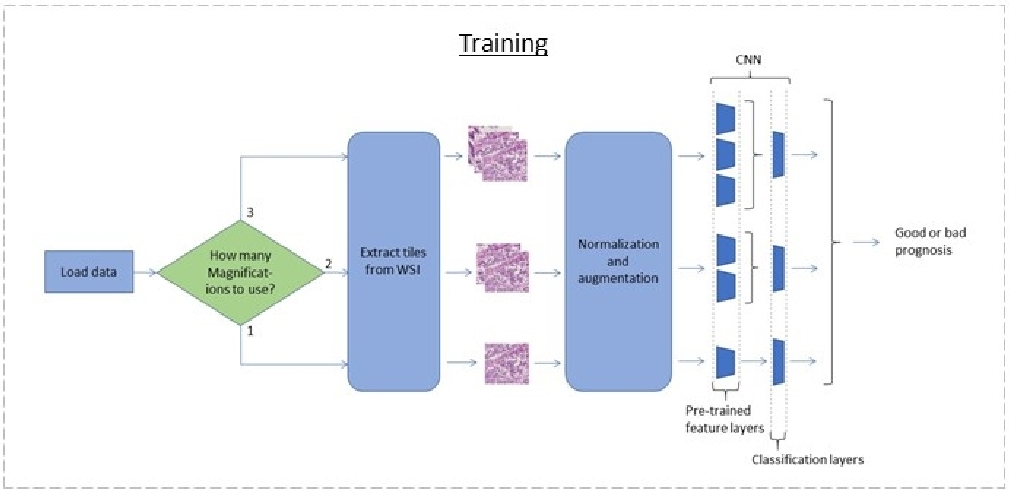
An image shown with RGB colors uses combinations of red, green and blue to make different colors, which means that a series of color combinations must be defined to represent a range of colors. Another method of representing colors in an image is through the HSV (**h**ue, **s**aturation, **v**alue) color model. This model is defined by:

- A color, hue, expressed as a number from 0 to 360 degrees [1].
- Amount of gray, saturation, in the color expressed as a number from 0 to 1 [2].
- Brightness, value, of the color expressed as a number from 0 to 1 [3].

[1] In python, cv2.inRange defines hue in range 0 to 180 degrees.
[2] In python, cv2.inRange defines saturation in range 0 to 255.
[3] In python, cv2.inRange defines value in range 0 to 255.



(a) Overview of preprocessing. Detailed figure shown in figure 5.2


Training
Load data
How many Magnificat-ions to use?
3
2
1
Extract tiles from WSI
Normalization and augmentation
CNN
Pre-trained feature layers
Classification layers
Good or bad prognosis

(b) Overview of training. Detailed figure shown in figure 5.4.

Figure 5.1: Pipeline for predicting bad or good prognosis, with tile(s) from an area going through the network.

An annotated lesion in a WSI contains background, showing the slide underneath the tissue. This background contributes to noise if used in a dataset. To remove this background, a tissue mask was made, as shown in figure 5.2, which extracted the blue, purple/pink color range, using the HSV format, to find every area affected by the H&E stain (explained in section 2.2).

Algorithm 1 was made to remove small regions and to close small holes to find all area in a tissue, without collecting small regions found outside the tissue. A threshold for removing small regions and close small holes was defined by variable *size2remove*, in algorithm 1. This algorithm was used to make the tissue mask, with *size2remove* set to 500, through trial. Hue was set to contain values from 100 to 180, which contain the blue and magenta color range [24] present in the tissue of H&E stained WSI, all saturation and brightness values were retained.

The annotated mask was made through creating and filling a polygon based on the annotation data from XML file, mentioned in section 4.2.1. The intersection of the tissue mask and the annotated mask results in a lesion mask, containing the area of a lesion, without background noise.

Annotating WSIs is a time consuming task, it is therefore useful to implement faster annotation methods. Figure 5.3 shows an alternative method of obtaining mask of lesions, where an auto segmentation, e.g., the auto segmentation method made by R. Amundsen [25], can be used instead of annotated data.

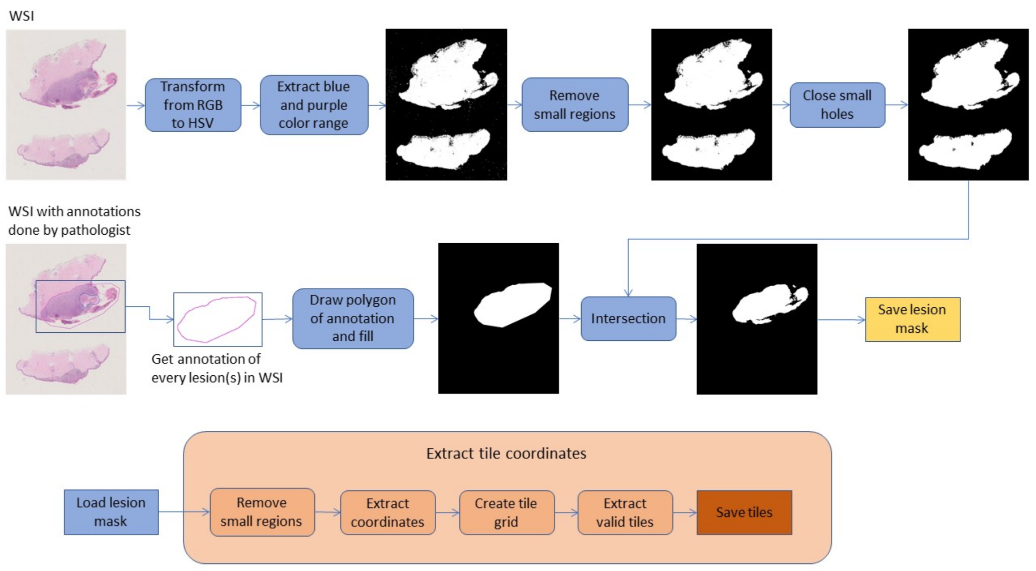


Figure 5.2: Overview of preprocessing. WSI and annotation data is used to make image mask consisting of areas in the annotated area, without the background noise. Function for extracting tile coordinates was made by Rune Wetteland [22].

Algorithm 1 Create lesion mask

```
Initialize:
    $Whole\ slide\ image, I$

/∗ Find blue and magenta colors in $I(x, y)$ /∗
$RGB\ to\ HSV \rightarrow I(x, y)_{HSV}$
$inRange((100, 0, 0), (180, 255, 255)) \rightarrow I(x, y)_{mask,HSV}$

$Remove\ small\ regions(size2remove = 500) \rightarrow I(x, y)_{mask,closed}$

/∗ Close small holes /∗
$Invert \rightarrow I(x, y)_{mask,inv}$
$Remove\ small\ regions(size2remove = 500) \rightarrow I(x, y)_{mask,inv,closed}$
$Invert \rightarrow I(x, y)_{mask}$

Result: $I(x, y)_{mask}$
```

Algorithm 2 Remove small regions

```
Inputs:
    $Image\ mask, Im$
    $size2remove, s2r$
$closing(footprint = square(3)) \rightarrow Im(x, y)_{closed}$
$Label\ connected\ regions \rightarrow label$
for each $region \in Measure\ properties\ of\ label$ do
    if $Area\ of\ region < s2r$ then
        $Im(Area\ of\ region)_{closed} = 0$
    end if
end for
Result: $Im(x, y)_{closed}$
```

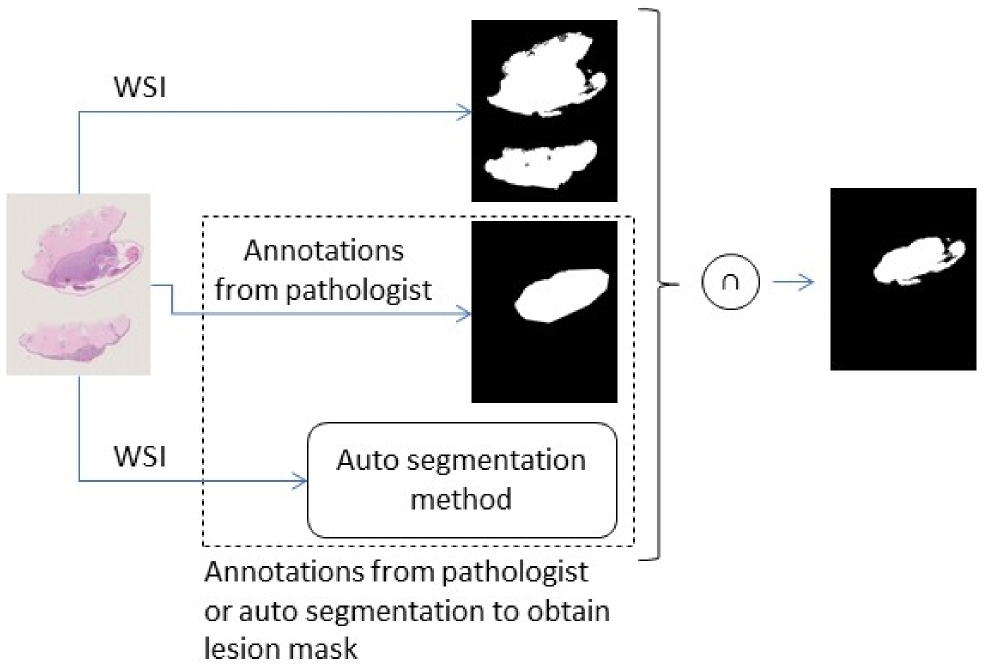

WSI
Annotations from pathologist
WSI
Auto segmentation method
Annotations from pathologist or auto segmentation to obtain lesion mask
∩

Figure 5.3: Method for obtaining mask of a lesion. An auto segmentation method can be used instead of manual annotations, to lessen workload.

5.2 Training and cross validation

Coordinates saved in the preprocessing part were used to make training and validation sets, using stratified 5-fold cross validation as shown in figure 5.4. One or multiple iterations of these folds were used to make batches of datasets, which were feed to a CNN. This CNN was trained for a predefined number of epochs, until the validation loss converged. A stochastic gradient descent optimizer, calculating the gradient decent with randomized selected samples, was used during training with 0.9 momentum. Early stopping was used to exit training if the validation loss did not get better over time. A new iteration was used if cross validation was utilized, and a new model was trained on the datasets from this iteration. Evaluation data and model from the best performing epoch(es) was saved.

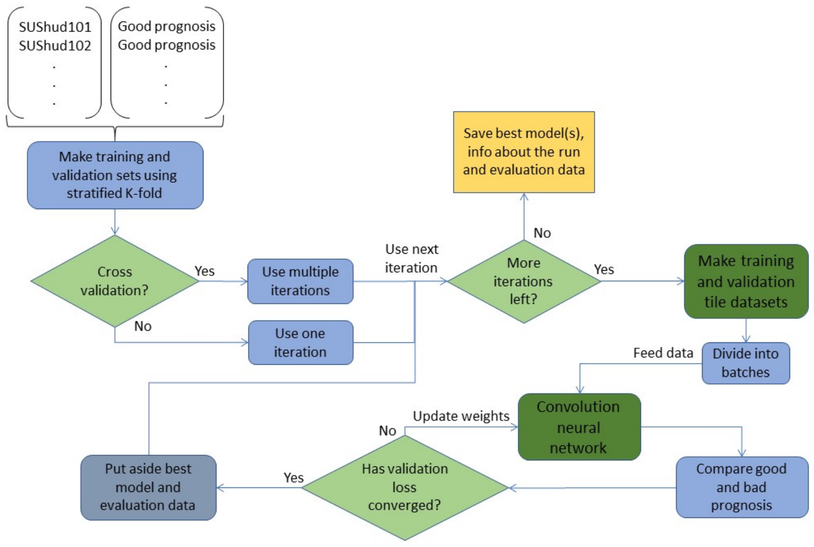

SUShud101
SUShud102
Good prognosis
Good prognosis
Make training and validation sets using stratified K-fold
Cross validation?
Yes
No
Use multiple iterations
Use one iteration
Use next iteration
More iterations left?
No
Yes
Save best model(s), info about the run and evaluation data
Make training and validation tile datasets
Divide into batches
Feed data
Convolution neural network
Update weights
No
Compare good and bad prognosis
Has validation loss converged?
Yes
Put aside best model and evaluation data

Figure 5.4: Overview of training. Function for making tile datasets and CNN shown in figure 5.5 and 5.6 respectively.

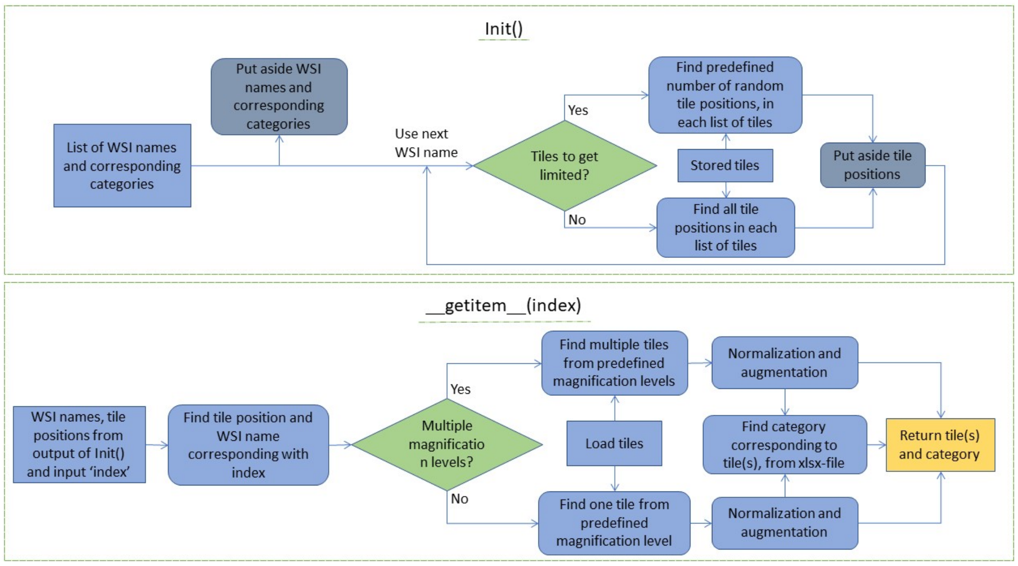
5.2.1 Making tile datasets

A dataset-method from PyTorch (*torch.utils.data.Dataset*) was used to let the CNN find data from the training set and validation set. Functions in this dataset-method is shown in figure 5.5, *Init()* and *__getitem__()*, which were changed to implement the data used in this thesis.

The *Init()* extracted a predefined number of start coordinates of tiles from each WSI from a dataset, in a randomized fashion. For WSIs with less coordinates then the predefined number, every coordinate was used. These coordinates were put aside, in addition to the number of coordinates extracted for each WSI, WSI names and corresponding categories.

__getitem__() extracts a tile from a WSI, referred to by the function's input index, using coordinates of tiles from the *Init()* function. These tiles are normalized using mean and standard deviation of images in ImageNet, to have tiles fit better with weights trained on ImageNet. Data augmentation is used, with random resize crop and random horizontal flip on the training set, to get variation in the training data. Train and validation set was resized from 256 to 224, for doing random resize crop and to have same size as the images in ImageNet.

Batches of train data and validation data, from the dataset-method, were made with a predefined batch size, using a data-loader from PyTorch (*torch.utils.data.DataLoader*). These batches were then feed to a CNN.



Init()
List of WSI names and corresponding categories
Put aside WSI names and corresponding categories
Use next WSI name
Tiles to get limited?
Yes
No
Find predefined number of random tile positions, in each list of tiles
Stored tiles
Find all tile positions in each list of tiles
Put aside tile positions
__getitem__(index)
WSI names, tile positions from output of Init() and input 'index'
Find tile position and WSI name corresponding with index
Multiple magnification levels?
Yes
No
Find multiple tiles from predefined magnification levels
Load tiles
Find one tile from predefined magnification level
Normalization and augmentation
Find category corresponding to tile(s), from xlsx-file
Normalization and augmentation
Return tile(s) and category

Figure 5.5: Method for making training and validation tile datasets. *Init()* finds random positions for a predefined amount of tiles, for each WSI in the dataset. *__getitem__(index)* finds a tile depending on the input index and returns the tile and it's category. *Init()* runs ones when the dataset is made, while the latter function runs every time an item is requested from the dataset.

5.2.2 CNN at multiple magnification levels

Pathologists examine malignant tissue at multiple levels to guess the prognosis of a patient. Mimicking this, multiple magnification levels were used in this thesis to predict prognosis of melanoma, as shown in figure 5.6. This was realized using CNNs, with two different methods: one for predicting single magnification level tiles, and one for predicting multiple magnification level tiles. A mono-scale model was defined as a model trained on one magnification level, di-scale model was defined as trained on two and tri-scale on three. All of these methods used the VGG16 network pre-trained on ImageNet as the initial feature extractor at each magnification level, with two neurons in the network's last FC layer. The feature extraction layers of the network were frozen to keep weights of basic features unchanged, leaving only the FC layers with trainable parameters. The number of trainable parameters were kept low because of the low number of patients at hand.

In the method with multiple magnification level tiles two or three magnification level tiles were each sent through unique feature extraction layers, as displayed in figure 5.6. These layers where concatenated and operated on by classification layers. An stochastic gradient descent (SGD) optimizer was used, calculating the gradient decent with randomized selected samples. Every single set of tile(s) going through the network resulted in a prediction of 0 or 1 (bad or good prognosis).

The result of the CNN was used to calculate prediction/evaluation data of the best performing epoch, with lowest validation loss. The model and prediction data of the best performing epoch was returned as outputs of the CNN function displayed in figure 5.4.

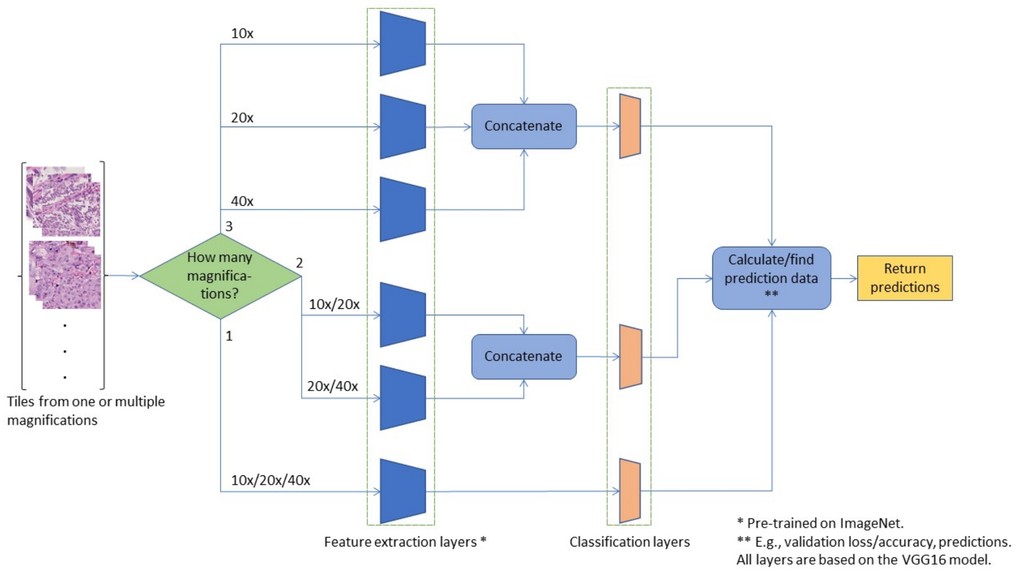

10x
20x
40x
3
How many magnifica-tions?
2
1
10x/20x
20x/40x
10x/20x/40x
Concatenate
Concatenate
Calculate/find prediction data **
Return predictions
Tiles from one or multiple magnifications
Feature extraction layers *
Classification layers
* Pre-trained on ImageNet.
** E.g., validation loss/accuracy, predictions.
All layers are based on the VGG16 model.

Figure 5.6: Overview of CNN architecture. One or multiple magnifications of a tile are fed as an input to the CNN and prediction data of the validation data are returned. Blue trapezoids represents frozen CNN layers and brown trapezoids represents FC layers with trainable layers. These trapezoids follows the VGG16 structure depicted in figure 3.7, with exception to the last FC layer, which in this function has two neurons.

5.3 Testing

There was not a test set to check the performance of the training, because of the small number of WSIs available. The result from validating on a validation set while training was therefore used as a test of the network's performance. The evaluation data used in this test was stored as text-files in the end of training the network, shown in figure 5.4.

Figure 5.7 shows overview of the evaluation data displayed to evaluate the network. The evaluation data contained validation and accuracy for every epoch, which was analyzed to see how well the network was training. The system was evaluated at tile-level and at WSI-level. When evaluating at tile-level, the WSI-label of good and bad prognosis was propagated down so that all tiles from that WSI were associated with that label. Prognosis at tile-level were used to find sensitivity, specificity and F_1 score. The prognosis at WSI-level was predicted using a threshold, where WSIs with proportion of predicted bad-prognosis tiles over or equal to the threshold, was predicted to have bad prognosis. The remaining WSIs was predicted to have good prognosis.

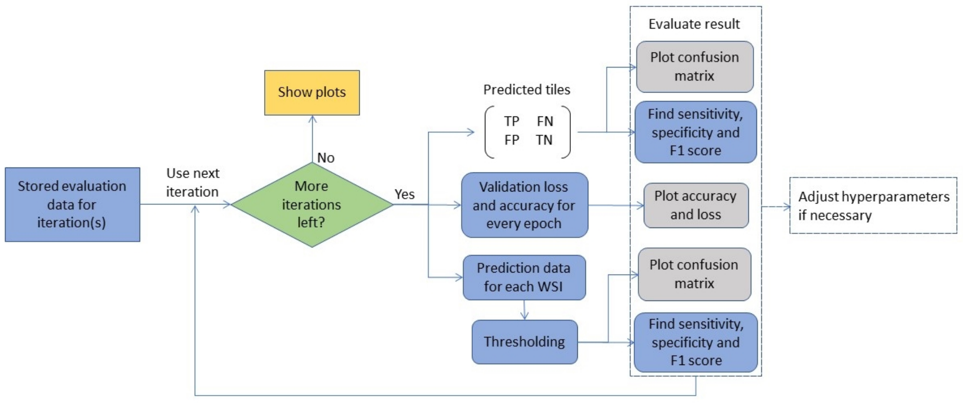

Evaluate result
Plot confusion matrix
Find sensitivity, specificity and F1 score
Plot accuracy and loss
Plot confusion matrix
Find sensitivity, specificity and F1 score
Show plots
Predicted tiles
TP FN
FP TN
No
Stored evaluation data for iteration(s)
Use next iteration
More iterations left?
Yes
Validation loss and accuracy for every epoch
Prediction data for each WSI
Thresholding
Adjust hyperparameters if necessary

Figure 5.7: Overview of evaluating validation data. The result is shown with values and plots, this is thereafter used to change hyperparameters and evaluate the network.

5.4 Implementation

The programming language *Python* was used to make the software for this thesis. Code produced by R. Wetteland [22] and N. Inkawhich [23] was combined with self-made code. *Python* combined with deep learning framework *PyTorch* was used for designing the CNN. Distribution of code, is displayed in table 5.1.

Python file	Function
OtherMethods.py	Extract tile coordinates.[1] Make mono-scale model.[2] Freeze layers.[2]
MyMethods.py	Make mask.[3] Make dataset.[3] Make multi-scale model.[3] Various other functions.[3]
Preprocessing.py	Uses functions to extract tile coordinates.[3]
TorchDNN.py	Setup for augmentation and normalization.[2] Optimization.[2] Train model.[2] Make training and validation history.[2] Collect prediction data.[3]
Prediction.py	Evaluate prediction data.[3]

[1] Made by R. Wetteland [22].
[2] Made by N. Inkawhich [23].
[3] Self made.

Table 5.1: Overview of python files and its functions, used in this thesis.

6. Experiments and results

This chapter presents experiments and results achieved in this thesis. The main objective of this thesis was to propose a method to predict good or bad prognosis of malignant melanoma. Experiments were done, narrowing down to a CNN model able to do this prediction with good results.

Results from preprocessing and results from the best performing models are presented, and the choice of parameters are discussed. Accuracy and loss of training and validation were used to evaluate model training, and sensitivity, specificity and F_1 score were used to evaluate model performance. Positive and negative labels in these metrics refers respectively to bad and good prognosis, as shown in figure 6.1. Correctly classifying bad prognosis was considered to be more essential then correctly classifying good prognosis, hence sensitivity was prioritized rather then specificity. Cross validation was used to evaluate dataset using different iterations, and receiver operating characteristic (ROC) curves were used to find suitable thresholds.

6.1 Experiment 1 - Preprocessing

Preprocessing of WSIs was done to find the location of lesions in the WSIs, using annotations made by co-supervisor H. Hardardottir, as explained in section 5.1. To find the location of a lesion, a tissue mask was made to separate the tissue from the background in a WSI, as illustrated in figure 5.2.

The preprocessing method was used on all 51 WSIs, where a predefined variable, named *size2remove*, was used to define the size of regions to remove and size of holes to fill. This variable was adjusted, through trail and error, to remove all regions belonging to the background of the WSIs, without removing parts of the WSIs belonging to a lesion. Through this trail and error, *size2remove* was set to 500, meaning that regions containing 500 pixels or less was removed, before holes containing 500 pixels or less was filled. The

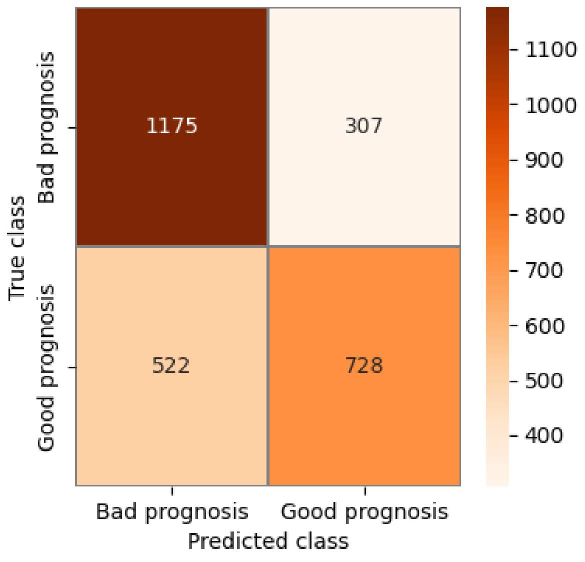
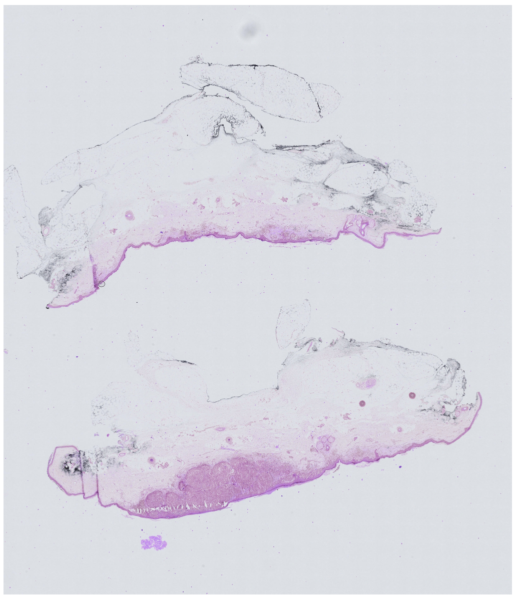


Figure 6.1: Confusion matrix showing results from using validation dataset $D_{v1}^{20,m20}$, with learning rate set to 0.0001. Class 1 (positive instances) refers to bad prognosis and class 2 (negative instances) refers to good prognosis.

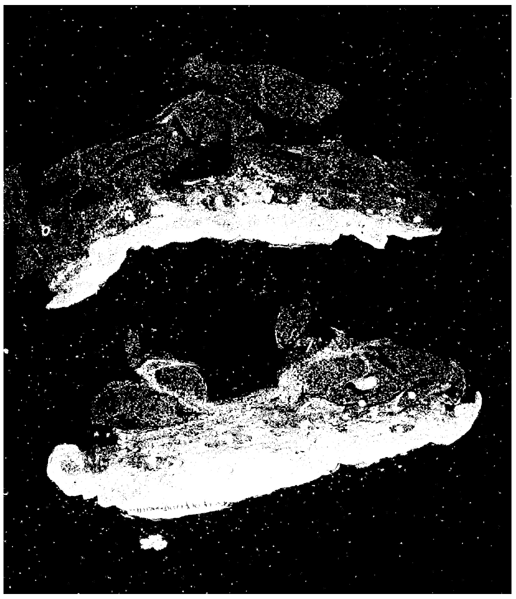
color range, hue, of WSIs in the HSV format was adjusted to find the colors left by H&E staining. The hue was first set to contain the blue and magenta colors, in range 120-180 [1] as defined in [24], the hue was thereafter changed through trail and error, to contain colors in range 100-180.

Experiments and examples of results is shown in figure 6.2 and 6.3. Areas removed by tissue masks and annotation masks is shown in figure 6.4, where remaining area contains a lesion, without background noise.

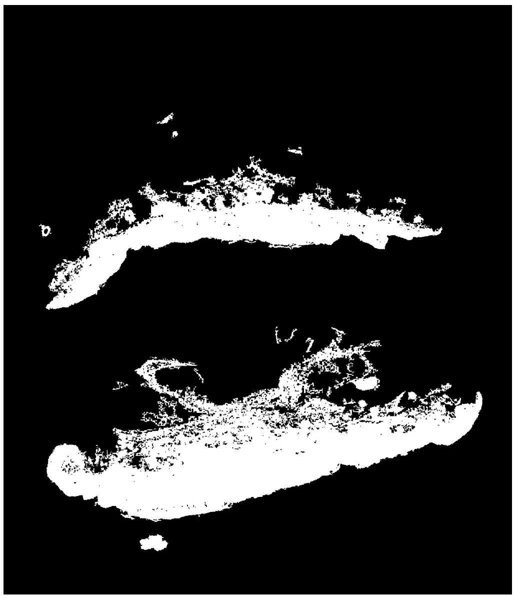
[1]Hue defined in range 0-180.



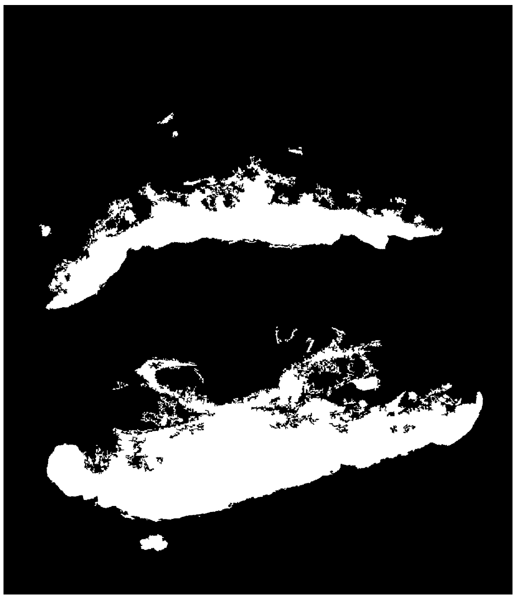
(a) Original image.



(b) Area with blue and magenta colors.

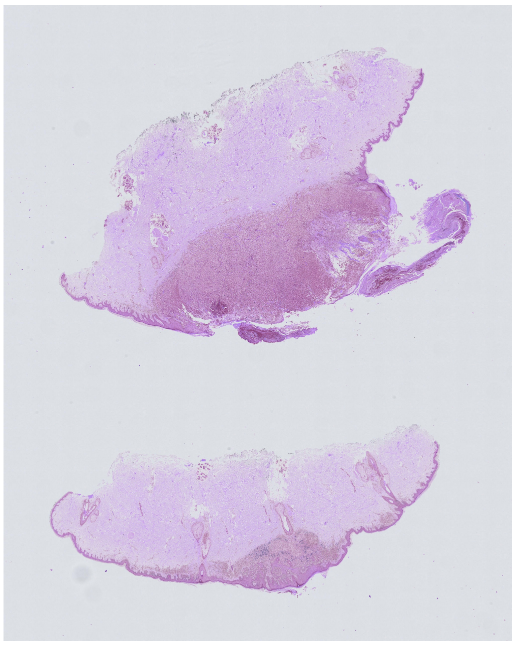


(c) Removed small regions.

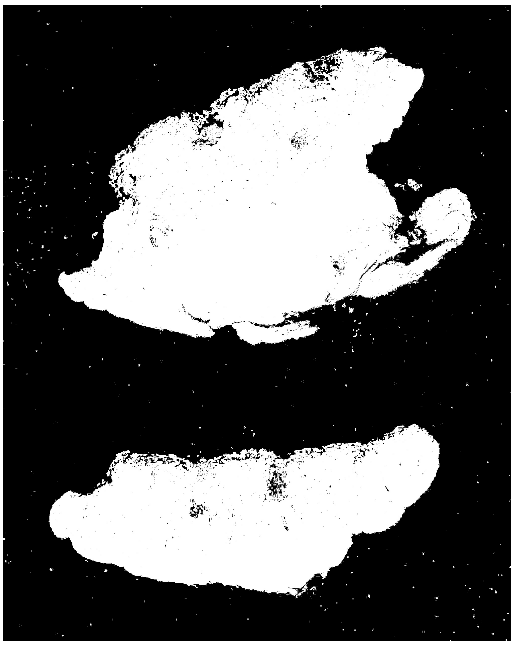


(d) Filled small holes. Final tissue mask.

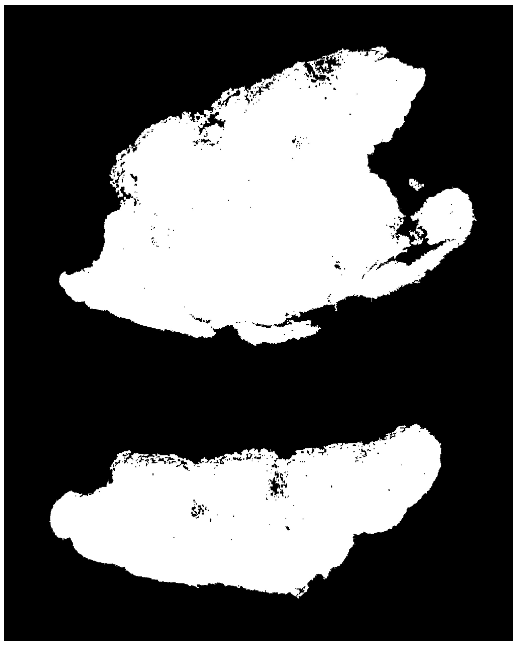
Figure 6.2: Process of making a tissue mask for a melanoma tissue with good prognosis. Blue and magenta colors of original image (6.2a) makes up mask in figure 6.2b. The small regions in this mask are removed (6.2c), before small holes are filled, resulting in the tissue mask in figure 6.2d.



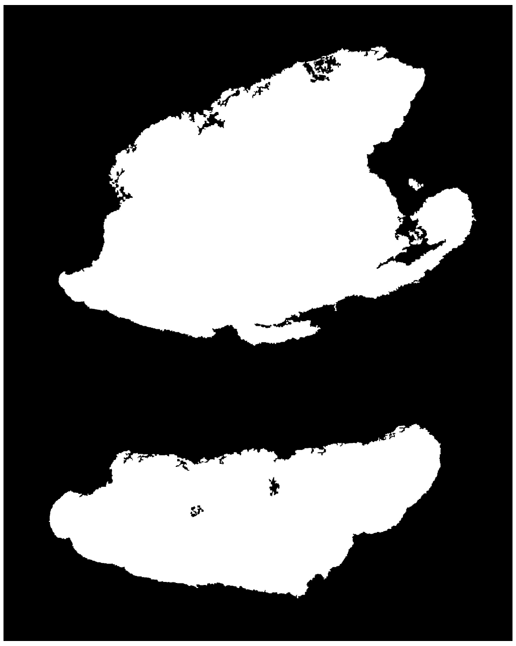
(a) Original image.



(b) Area with blue and magenta colors.



(c) Removed small regions.



(d) Filled small holes. Final tissue mask.

Figure 6.3: Process of making a tissue mask for a melanoma tissue with bad prognosis. Blue and magenta colors of original image (6.3a) makes up mask in figure 6.3b. The small regions in this mask are removed (6.3c), before small holes are filled, resulting in the tissue mask in figure 6.3d.

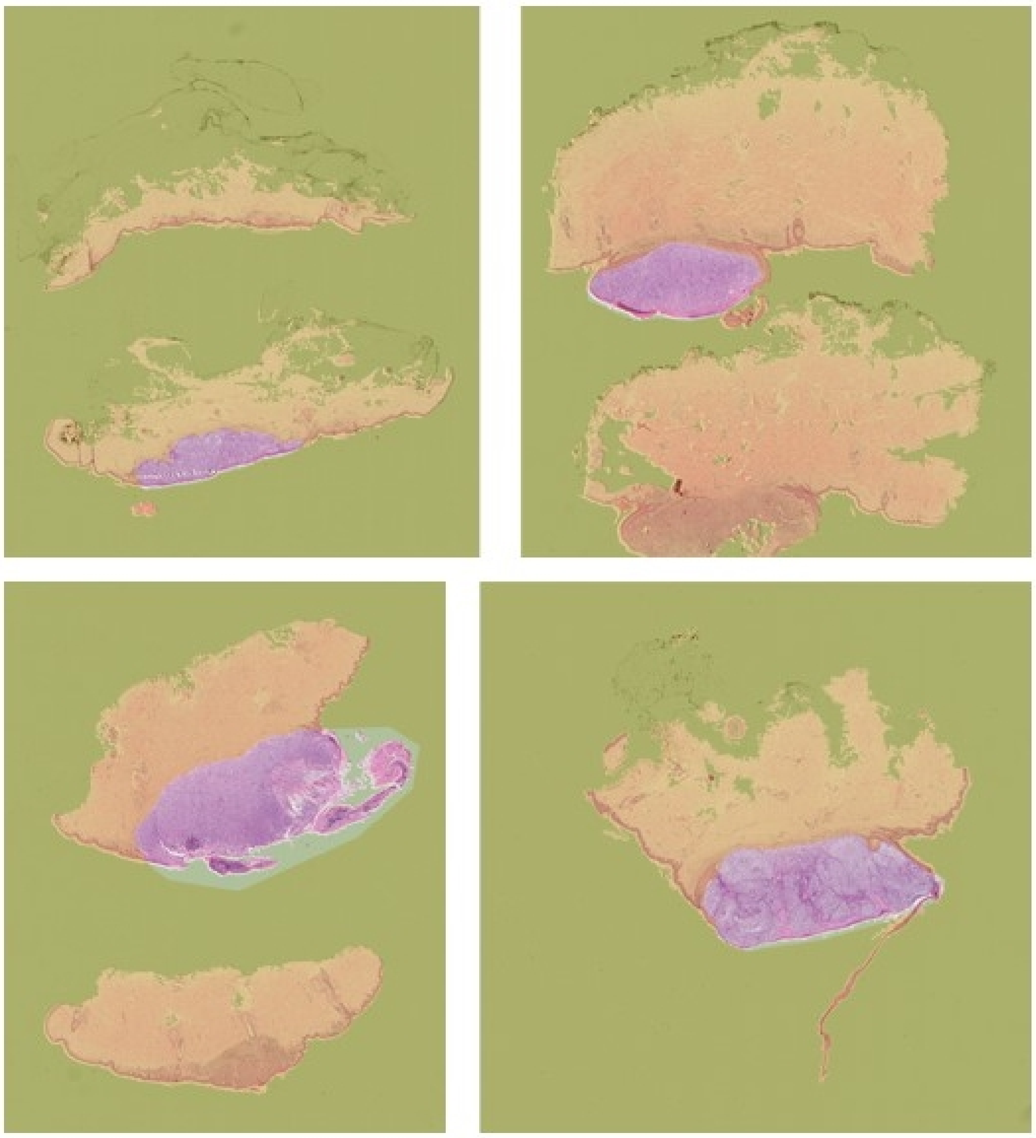


Figure 6.4: Examples of tissue masks and annotated masks, showing lesion in the WSIs, with top images having good prognosis and bottom images having bad prognosis. The area shaded with green is outside of tissue mask and area shaded with yellow/orange is outside of annotated mask. Olive green shade is area outside of tissue mask and annotated mask. The area inside both masks contains the lesion.

6.2 Experiment 2 - Finding optimal tile magnification

Experiments were done to find a optimal tile magnification for training a model. In these experiments, magnification level 20x or 40x were used as reference for tile extraction, and $m10$, $m20$, $m40$, mDi or $mTri$ were used for training (variables explained in table 4.1). $m10$, $m20$ and $m40$ had on magnification for every data input and was trained on a mono-scale model, mDi had two and was trained on a di-scale model and $mTri$ had three and was trained on a tri-scale model. Models were trained using a learning rate of 0.001 or 0.0001, and trained until their validation loss converged. The model weights at the epoch with lowest validation loss were saved, in addition to the model's prediction/evaluation data. Early stop callback for 10 epochs was used to stop the model from overfitting. An overview of parameters used in the experiments is shown in table 6.1.

The WSIs in the dataset had different sized lesions, which led to a significant difference in the number of tiles extracted from each WSI, as shown in figure A.1 and A.2. This, along with long model training time, prompted the use of a tile limit, limiting the number of tiles extracted from each WSI to 250 tiles.

Mono-scale models converged relatively fast, so the epoch limit for these models was set to 15 epochs. The models with two magnifications took longer to converge, so the epoch limit for these models was set to 22 epochs.

6.2.1 Experiment 2a

The objective of this experiment was to find a magnification level fit to be used in a model. All magnification levels (mj) was used in this experiment, and magnification level 20x was used as reference, with learning rate set to 0.001.

Figure 6.2 shows performance from predicting prognosis of tiles, with F_1 scores well over 0.5 for every dataset. These datasets, except for $D_{v1}^{20,mTri}$, results in the model having sensitivity close to 0.8 and specificity over 0.5.

Looking at the accuracy and loss of the models in figure 6.5, one can see that figures 6.5a to 6.5c refer to models that are hardly training, and figure 6.5d and 6.5e refers to models with validation loss that is failing to converge.

Parameter	Changed	Variation
Reference magnification level	Yes	20x or 40x magnification level.
Extracted magnification level(s)	Yes	$m10$, $m20$, $m40$, mDi or $mTri$, as described in table 4.1.
Learning rate	Yes	0.001 or 0.0001.
Epoch limit	Yes	15 or 22, depending on how fast the model converged.
Tile limit	No	250 tiles per WSI.
Data split	No	One stratified 5-fold iteration as validation data, rest as training data.
Patient	No	10 epochs.
Batch size	No	32.
Data augmentation	No	Random resize crop and random horizontal flip in the training set.
Dropout rate	No	50%.
Optimizer	No	SGD with 0.9 momentum.
Network layers to update	No	Updates only classification layers.

Table 6.1: Overview of training parameters, utilized in the experiments of this thesis.

The accuracy and loss shown in the last mentioned figures suggests that the model has too large of a complexity and/or learning rate, as discussed by J. Brownlee in his book about better deep learning [26].

The next experiments were done with a lower learning rate, to improve model training.

6.2.2 Experiment 2b

The datasets used in the experiment in section 6.2.1 where used in this experiment, with a learning rate set to 0.0001.

Figure 6.3 shows performance from predicting prognosis of tiles, with F_1 higher than 0.7 for $D_{v1}^{20,m20}$, $D_{v1}^{20,m40}$ and $D_{v1}^{20,mDi}$. The sensitivity is around 0.8 for these datasets, with it being over 0.85 for $D_{v1}^{20,mDi}$, and the specificity

Architecture	Validation dataset	Sensitivity	Specificity	F_1 score
Mono	$D_{v1}^{20,m10}$	**0.7989**	0.5088	0.7220
	$D_{v1}^{20,m20}$	0.7895	**0.5512**	**0.7283**
	$D_{v1}^{20,m40}$	0.7740	0.5408	0.7162
Di	$D_{v1}^{20,mDi}$	0.7949	0.5176	0.7220
Tri	$D_{v1}^{20,mTri}$	0.6430	0.6320	0.6584

Table 6.2: Results from experiment 2a, showing performance from predicting prognosis of tiles in validation sets.

is just under 0.6 for $D_{v1}^{20,m20}$.

The accuracy and loss shown in figure 6.6 shows that the models, using these datasets are able to learn from the datasets, which results in the validation loss converging. This is especially visible when using $D_{v1}^{20,mDi}$, where validation loss drops from 0.7 to 0.6.

Architecture	Validation dataset	Sensitivity	Specificity	F_1 score
Mono	$D_{v1}^{20,m10}$	0.6916	0.5576	0.6699
	$D_{v1}^{20,m20}$	0.7928	**0.5824**	**0.7392**
	$D_{v1}^{20,m40}$	0.8205	0.4552	0.7197
Di	$D_{v1}^{20,mDi}$	**0.8596**	0.4440	0.7383
Tri	$D_{v1}^{20,mTri}$	0.7072	0.5720	0.6838

Table 6.3: Results from experiment 2b, showing performance from predicting prognosis of tiles in validation sets.

6.2.3 Experiment 2c

In this experiment, datasets with $m20$, $m40$ and mDi were used, because of fewer details in $m10$, contained in $mTri$, and inferior F_1 score of $m10$ and $mTri$ in experiment shown in section 6.2.2. Magnification level 40x was used as reference and the learning rate was set to 0.0001.

Learning and validation curves of this experiment is shown in figure 6.4, showing a sensitivity under 0.4 when using $D_{v1}^{40,m20}$ and specificity close to zero for $D_{v1}^{40,mDi}$. The sensitivity is over 0.6 when using $D_{v1}^{40,m40}$, but the specificity for using this dataset is close to 0.4.

Figure 6.7 shows no improvement in loss or accuracy when training on the datasets, leading to early stopping at epoch 10.

Architecture	Validation dataset	Sensitivity	Specificity	F_1 score
Mono	$D_{v1}^{40,m20}$	0.3506	**0.6339**	0.4292
Di	$D_{v1}^{40,m40}$	**0.6407**[1]	0.4134	**0.6117**[1]
Tri	$D_{v1}^{40,mDi}$	0.9953	0.0052	0.7199

[1] Sensitivity and F_1 score are highlighted for $D_{v1}^{40,m40}$ because of the terrible specificity of $D_{v1}^{40,mDi}$.

Table 6.4: Results from experiment 2c, showing performance from predicting prognosis of tiles in validation sets.

6.3 Experiment 3 - Evaluation with cross validation

Correctly classifying bad prognosis was considered to be more essential then correctly classifying good prognosis, hence high sensitivity was seen as more important then high specificity. Cross validation was done on models trained on $D_{t/vk}^{20,m20}$ and $D_{t/vk}^{20,mDi}$, from experiment 2b in section 6.2, because their combination of superior F_1 score and high sensitivity. Cross validation was used to find a better estimate of how well these two models performed. Sensitivity, specificity and F_1 score was compared for each iteration of a dataset, to see if these metrics correlated. Accuracy and loss for training and validation in a model were compared, to observe if the model was training in each iteration of the dataset.

In this section, prediction of tiles in WSIs, labeled with the same category as corresponding WSI, were found. These predictions where then used to observe the network's capability to recognize tiles from each prognosis.

Experiments was done to classify each WSI as good or bad prognosis, to find the network's capability to classify the WSIs. In these experiments a threshold was used for the classification of WSIs, as described in section 5.3, which was decided by the use of the model's ROC. The ROC showed the sensitivity plotted over 1-specificity, which was calculated using multiple thresholds. Area under the curve of a ROC (AUC) was used as an indication of a model's performance and should be as close to 1 as possible. Poor values of AUC were in the range 0.6-0.7, while good values were over 0.8 [27]. Accuracy was used to compare models from these experiments, with models from related work.

The model utilized in the following experiments had a learning rate set to 0.0001, and used different iterations of datasets $D_{t/v1}^{20,m20}$ and $D_{t/v1}^{20,mDi}$, which was used in the experiment in section 6.2.2. Results of the experiment in section 6.2.2 are shown in table 6.3, with training performance shown in figure 6.6a and 6.6c.

6.3.1 Experiment 3a - Mono-scale model

The datasets used in this experiment were $D_{t/vk}^{20,m20}$, with iterations in range 1 to 5.

Table 6.5 shows performance from classifying tiles, where the sensitivity and F_1 score of using the iterations, is for most of them over 0.6. The highest sensitivity and F_1 score was achieved by using iteration 1, which also gave the lowest specificity, compared to using the other iterations.

Architecture	Iteration	Sensitivity	Specificity	F_1 score
Mono	1	0.7935	0.5496	0.7302
	2	0.6032	0.5689	0.6153
	3	0.6152	0.6993	0.6570
	4	0.6344	0.6728	0.6619
	5	0.5551	0.5566	0.5578

Table 6.5: Results from experiment 3a, showing performance from predicting prognosis of tiles in iterations, using $D_{t/v1-5}^{20,m20}$.

Figure 6.8 shows accuracy and loss for training and validation of the model. The model was not stopped by early stopping, but for some of the iterations the validation loss was not distinctly decreasing.

The threshold for classifying WSIs in the dataset was decided by using a ROC curve, shown in figure 6.9 with an AUC of 0.81. A detailed table of this ROC, shown in table B.1, was used to decide a threshold of 0.3720. This threshold was chosen because of its corresponding, relatively high sensitivity, and specificity over 0.5 (1-specificity was under 0.5). Results from classifying WSIs is shown in figure 6.6, where most of the iterations resulted in a sensitivity equality to, or over 0.8 and a F_1 score higher then 0.76. The specificity for most of the iterations is 0.76 or higher.

Architecture	Iteration	Sensitivity	Specificity	F_1 score	Accuracy
Mono	1	1.0	0.6	0.8571	0.8182
	2	1.0	0.4	0.7692	0.7
	3	0.6	0.8	0.6667	0.7
	4	1.0	0.8	0.9091	0.9
	5	0.8	0.2	0.6154	0.5
Total:		0.8846	0.4400	0.7667	0.7255

Table 6.6: Results from experiment 3a, showing performance from predicting prognosis of WSIs in iterations, using $D^{20,m20}_{t/v1-5}$. Threshold set to 0.3720 and AUC = 0.81.

6.3.2 Experiment 3b - Multi-scale model

The datasets used in this experiment were $D^{20,mDI}_{t/vk}$, with iterations in range 1 to 5.

The performance from classifying tiles is shown in table 6.7, with a sensitivity and F_1 score over 0.55, when using most of the iterations in the dataset. Specificity is close to zero for iteration 2 and higher than 0.5 for most of the other iterations.

Accuracy and loss of training and validation are shown in figure 6.10, where the training on most of the iterations, results in a validation loss that converges to an lower value. Training on iteration 2 and 5, shown in figure 6.10b and 6.10d, results in no significant decrease in validation loss.

Figure 6.11 shows the ROC curve used to find a threshold to classify WSIs, with an AUC of 0.68. This threshold was chosen to be 0.4480, from the detailed table of the ROC curve, shown in table B.3. This threshold was

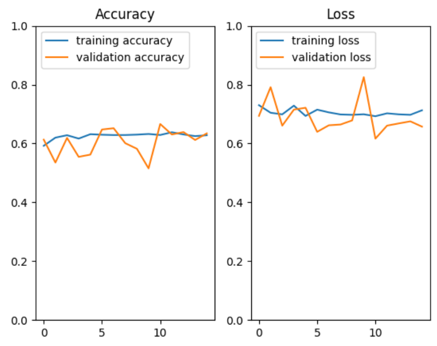
Architecture	Iteration	Sensitivity	Specificity	F_1 score
Di	1	0.7605	0.5392	0.7077
	2	0.8824	0.0906	0.6693
	3	0.6864	0.5826	0.6716
	4	0.4640	0.8526	0.5832
	5	0.5613	0.4805	0.5420

Table 6.7: Results from experiment 3b, showing performance from predicting prognosis of tiles in iterations, using $D^{20,mDi}_{t/v1-5}$.

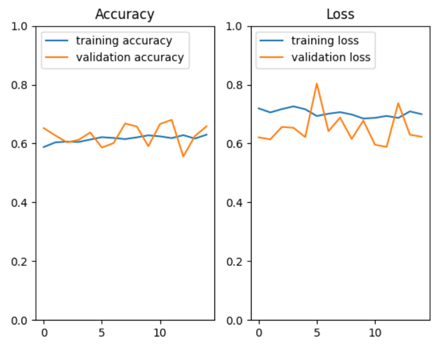
chosen because it corresponded to a relatively high sensitivity, with specificity over 0.5 (1-specificity was under 0.5). The results from predicting prognosis of WSIs is shown in table 6.8, where most of the iterations resulted in a sensitivity equal to, or over 0.8 and a specificity equal to, or over 0.66.

Architecture	Iteration	Sensitivity	Specificity	F_1 score	Accuracy
Di	1	1.0	0.6	0.8571	0.8182
	2	1.0	0.0	0.6667	0.5
	3	0.6	0.6	0.6	0.6
	4	0.6	1.0	0.75	0.8
	5	0.8	0.4	0.6667	0.6
Total:		0.8077	0.4800	0.7119	0.6667

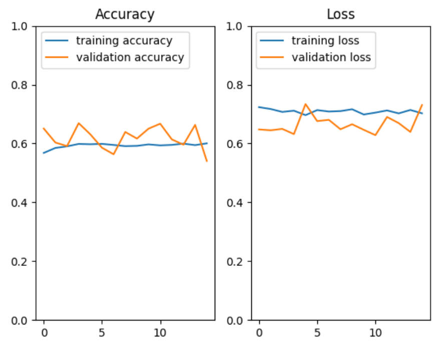
Table 6.8: Results from experiment 3b, showing performance from predicting prognosis of WSIs in iterations, using $D^{20,mDi}_{t/v1-5}$. Threshold set to 0.4480 and AUC = 0.68.



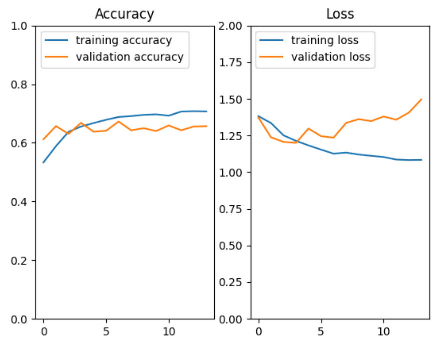
(a) $D^{20,m10}_{t/v1}$.



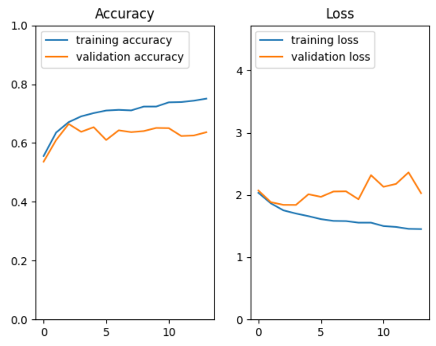
(b) $D^{20,m20}_{t/v1}$.



(c) $D^{20,m40}_{t/v1}$.

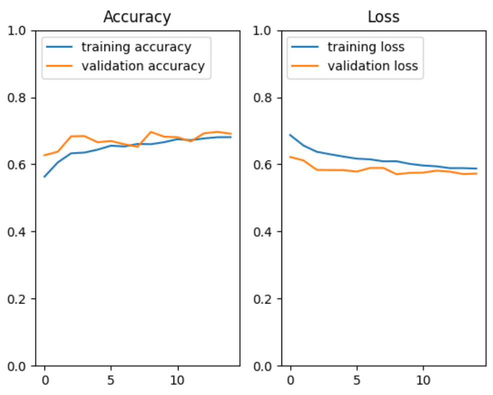


(d) $D^{20,mDi}_{t/v1}$.

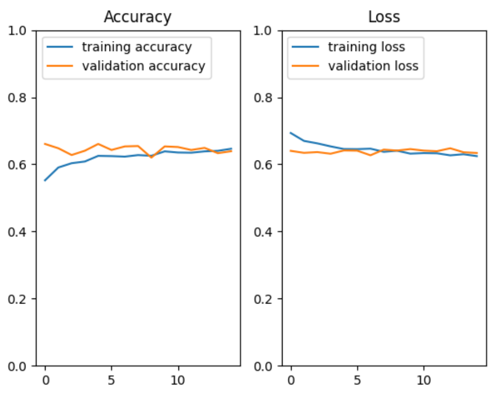


(e) $D^{20,mTri}_{t/v1}$.

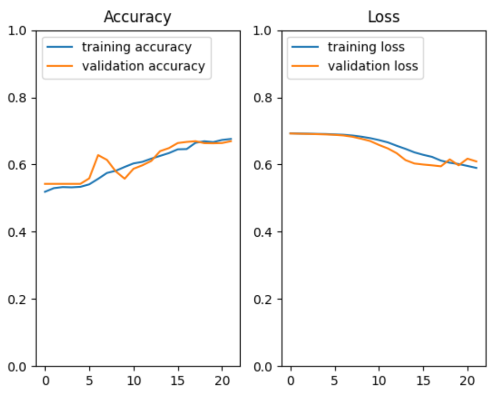
Figure 6.5: Accuracy and loss of training and validation over epochs, from experiment 2a.



(a) $D_{v1}^{20,m20}$.

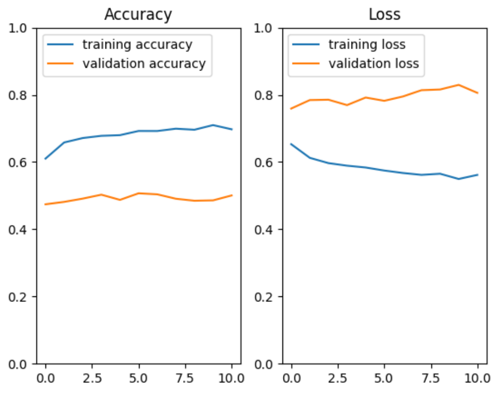


(b) $D_{v1}^{20,m40}$.

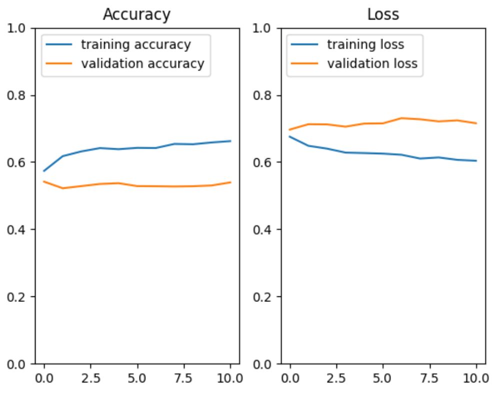


(c) $D_{v1}^{20,mDi}$.

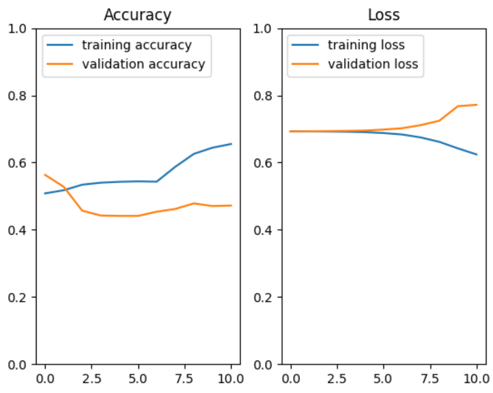
Figure 6.6: Accuracy and loss of training and validation over epochs, from experiment 2b.



(a) $D_{v1}^{40,m20}$.

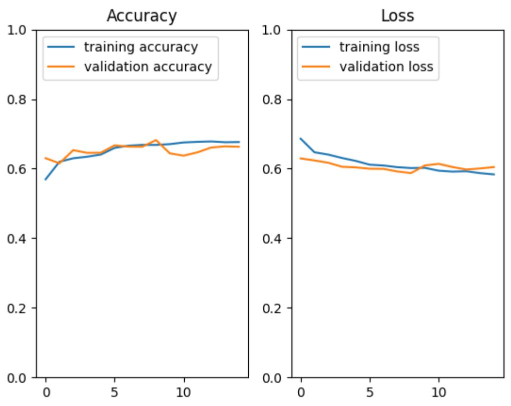


(b) $D_{v1}^{40,m40}$.

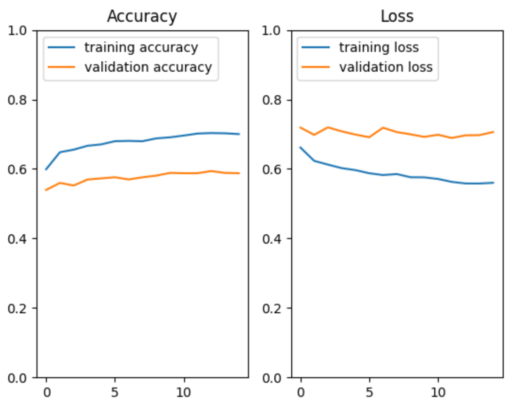


(c) $D_{v1}^{40,mDi}$.

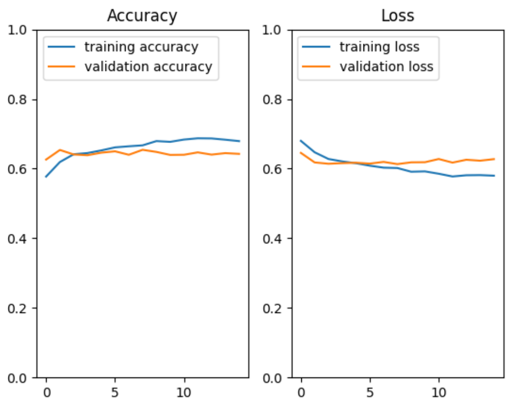
Figure 6.7: Accuracy and loss of training and validation over epochs, from experiment 2c.



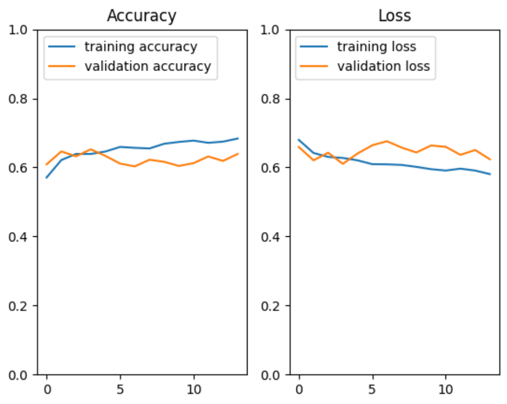
(a) Iteration 1.



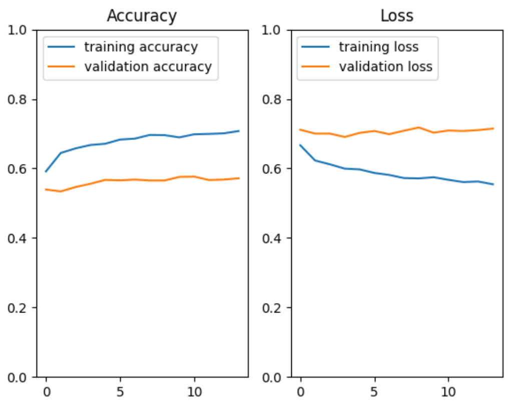
(b) Iteration 2.



(c) Iteration 3.



(d) Iteration 4.



(e) Iteration 5.

Figure 6.8: Accuracy and loss of training and validation over epochs, from experiment 3a, using cross validation with $D^{20,m20}_{t/v1-5}$.

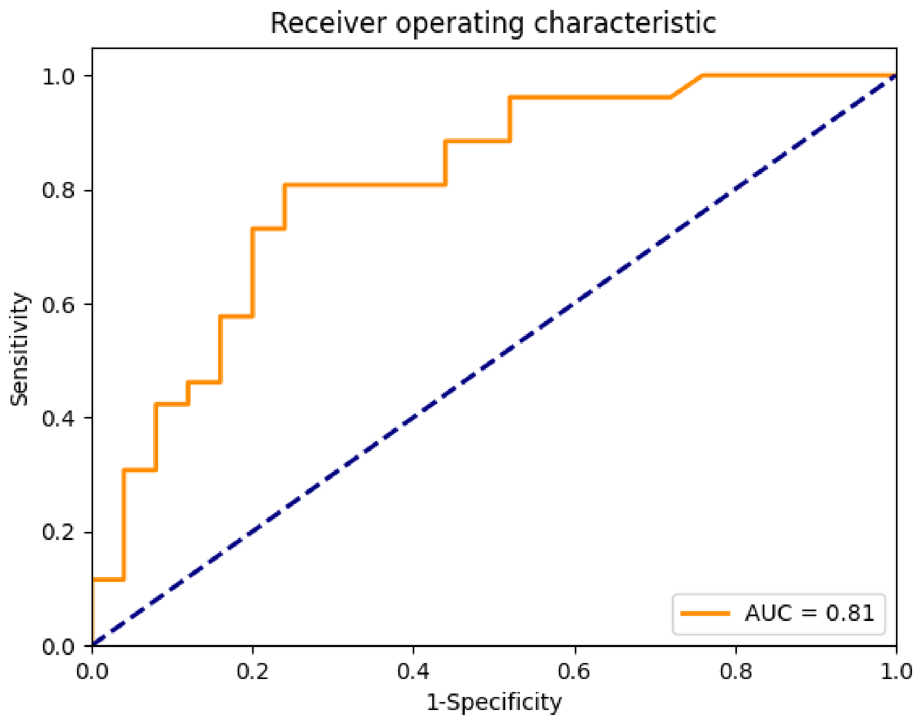
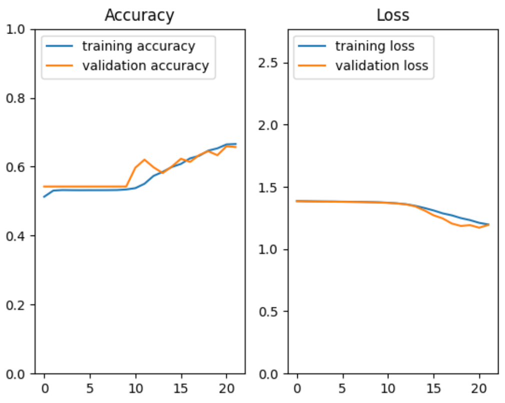
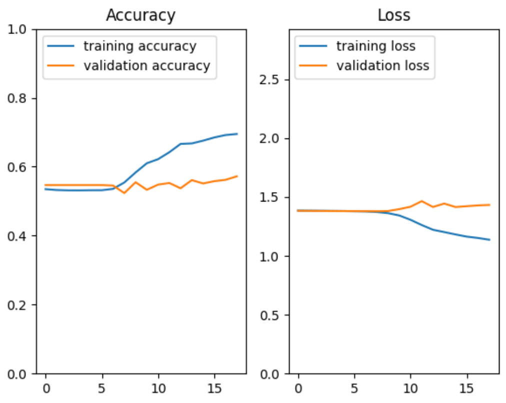


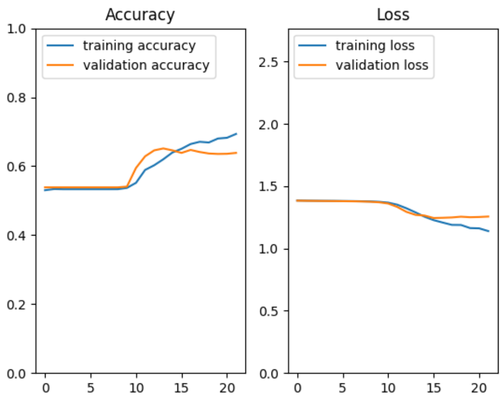
Figure 6.9: ROC curve and AUC from experiment 3a, using iterations from $D^{20,m20}_{t/v1-5}$.



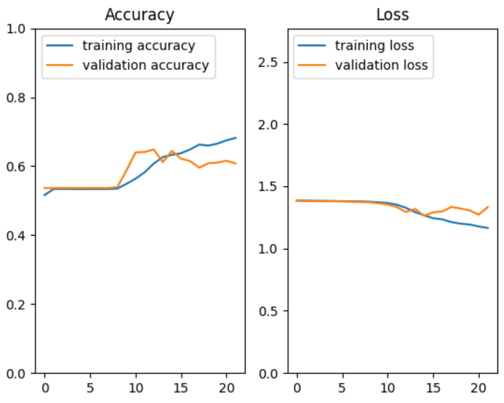
(a) Iteration 1.



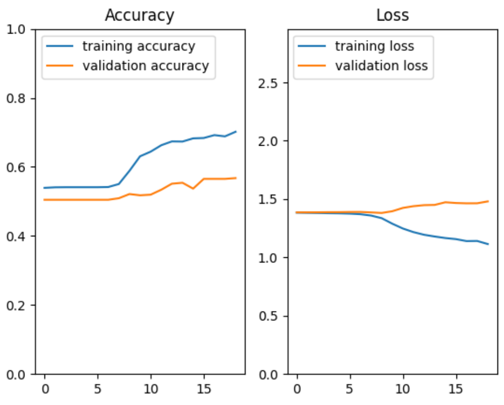
(b) Iteration 2.



(c) Iteration 3.



(d) Iteration 4.



(e) Iteration 5.

Figure 6.10: Accuracy and loss of training and validation over epochs, from experiment 3b, using cross validation with $D^{20,mDi}_{t/v1-5}$.

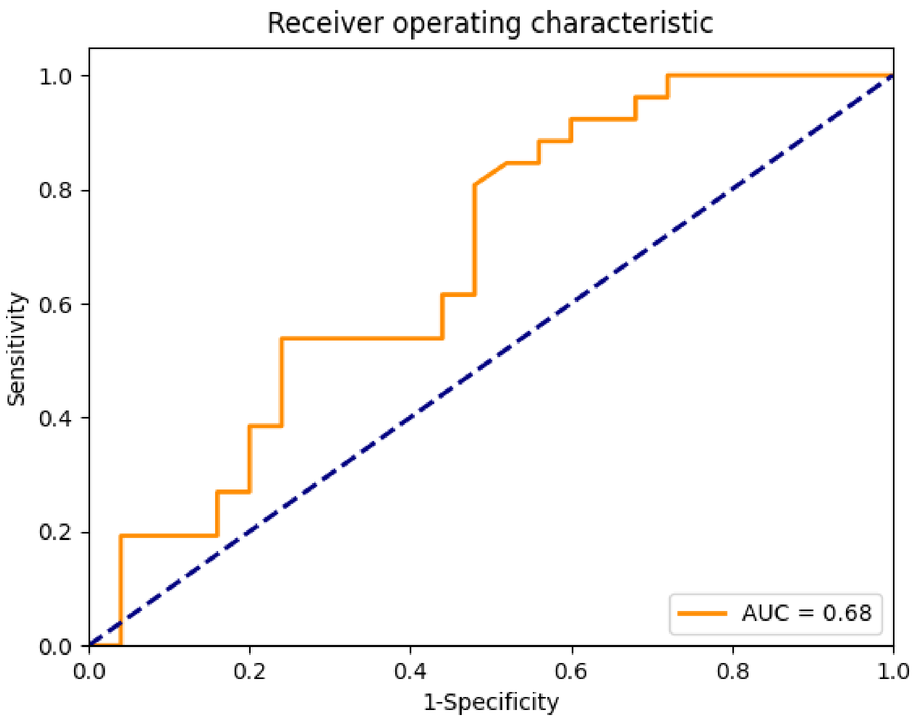


Figure 6.11: ROC curve and AUC from experiment 3b, using iterations from $D_{t/v1-5}^{20,mDi}$.

7. Discussion

This chapter gives a review of chapter 6. Achieved results, comparisons with related work, and limitations of data are discussed. Suggestions for future work is also presented.

7.1 Experiment 1 - Preprocessing

In the preprocessing, the HSV color-map was found to be useful for extracting colors in WSIs, left by H&E staining. The tissue masks proved useful for extracting the tissues in the WSIs, leaving out the background.

Even though the tissue masks extracted a lot of tissue, these masks removed parts of the tissues consisting of fat, because of the low concentration of cells in these areas. This could be a problem in cases where these areas were interesting, but did not cause any problem in this experiment, because of the high concentration of cells in lesions.

Figure 6.3 shows that large holes in the tissue (bottom tissue in 6.3a) are removed by the final tissue mask, but smaller holes (top tissue in 6.3a) are not removed. This leads to some of the holes being present in the extracted tiles, which can cause more background, then initially intended, to be present in the tiles.

7.2 Model performance

This section summarizes and discusses the experiments and results achieved in chapter 6. The dataset format used in this section is explained in table 4.1.

7.2.1 Experiment 2 - Utilizing different tile magnifications

The models showed good performance when using learning rate set to 0.001, but an increase in model learning when utilizing a smaller learning rate. Models trained on $D_{v1}^{20,mj}$, with learning rate set to 0.0001 showed good results when using $m20$, $m40$ and mDi. Models trained on $D_{v1}^{40,mj}$, with the same learning rate showed worse results.

The models using $D_{v1}^{20,m20}$ and $D_{v1}^{20,mDi}$ where considered as the most optimal models, because of their good sensitivity and F_1 score. Nevertheless, the experiments showed that utilizing multiple iterations of a dataset conveyed a better overview of model performance, and could influence the choosing of an optimal model.

7.2.2 Experiment 3 - Utilizing cross validation

Predicting prognosis of tiles in the mono-scale model showed a good prognosis in most of the iterations. Predicting prognosis of WSIs showed a good AUC of 0.81, and a good F_1 score and accuracy of 0.7667 and 0.7255, respectively, when using a threshold set to 0.3720. A better F_1 score of 0.7812 can be achieved by lowering the threshold to 0.3320, resulting in a higher sensitivity (from 0.8846 to 0.9615), but a lower specificity (from 0.56 to 0.48) and no change in accuracy, as shown in figure B.1. Furthermore, a F_1 score of 0.7925 can be achieved by increasing the threshold to 0.5280, resulting in a higher specificity (from 0.56 to 0.76) and accuracy (from 0.7255 to 0.7843), at a cost of lower sensitivity (from 0.8846 to 0.8077).

The performance of predicting prognosis of tiles in the multi-scale model, showed promising results in the first and fifth iterations, because of high sensitivity, but bad results in the rest because of low sensitivity or very low specificity. Predicting prognosis of WSIs showed a poor AUC of 0.68, and a F_1 score and accuracy of 0.7119 and 0.6667, respectively, with the threshold set to 0.4480. A better F_1 score could be achieved by lowering the threshold, resulting in higher sensitivity, but lower specificity and lower, or same accuracy, as seen in figure B.3.

The mono-scale model showed better performance than the multi-scale model. Nevertheless, different hyperparameters could prove better result for the multi-scale model. The best mono-scale model had an AUC equal to 0.81

and a F_1 score equal to 0.7667, with the threshold set to 0.3720. The best multi-scale model had an AUC equal to 0.68 and a F_1 score equal to 0.7119, with the threshold set to 0.4480.

7.3 Comparisons with related work

Results from experiments, were compared to papers about prognosis prediction of other cancers then melanoma. These papers predicted 5-year survival rate of non-small cell lung cancer (NSCLC) and breast cancer. AUC and accuracy were used for the comparisons.

Lai et al. [6] reported a result of 0.8163 AUC and 0.7544 accuracy, predicting prognosis of NSCLC by using a di-scale DNN model trained on biomarkers and clinical data from 614 patients. Zhang et al. [7] reported a result of 0.8508 AUC and 0.8096 accuracy, predicting prognosis of NSCLC by using a feature transformation method based on CNN. The data used in this model consisted of 331 patients.

Sun et al. [8] reported a result of 0.845 AUC and 0.826 accuracy, predicting prognosis of breast cancer, using a tri-scale DNN model with three datasets of medical data from 1 980 patients. Cheng et al. [9] reporting a result of 0.7836 AUC and 0.7179 accuracy, using a di-scale DNN model combined with ensemble learning to predict prognosis of breast cancer. The data used in this model consisted of 582 patients.

The best model produced in this thesis gave an AUC of 0.81 and an accuracy of 0.7255. The best comparisons to this result might been the research done by Lai et al. and Sun et al., because of the use of multi-scale models. This comparisons could not be drawn directly, because of different datasets.

7.4 Limitations

The dataset in this thesis contained of 51 WSIs from 51 patients, which was considered a small dataset for training a DNN, with increased probability for overfitting. Transfer learning was therefore used to obtain the basic features of images, and augmentation was utilized on the training set to expose the network for new variants of the WSIs.

A train dataset and a validation dataset was used to train and validate the network, and cross-validation was used to test the network. This could lead to biased results, because the validation dataset, used in the decision of network parameters, was used to test the network. A test dataset was not retain for testing because of the small number of patients available.

The tiles extracted from WSIs were randomly selected, to represent the WSIs as best as possible. This could lead to insecurity in the result, if tiles with negligible features were chosen to represent a lesion, but the cross validation showed good results, indicating that the network was able to find features to correctly predict WSI prognosis.

7.5 Future work

The number of data available is a essential issue when training a DNN. Retrieving more data leads to more features that the network can learn from. Experiments with other augmentation methods can be done to introduce new variations of the existing data, in both the training and validation dataset.

Experiments with other CNN architectures, then VGG16, should be performed to possibly create better models, and a wider range of hyperparameters can be utilized for locating better learning curves.

A method for locating clusters of tiles can be developed, to see if tiles groups up based on their corresponding predicted prognosis. This clustering method can possibly improve the prediction performance and can be useful for pathologists who analyze melanoma tissues.

Auto segmentation can be utilized to automatically locate the lesions in the WSIs, leading to less work for the pathologists.

8. Conclusion

The objective of this thesis was to propose a method for predicting prognosis of melanoma. Traditional image processing techniques and annotated data from a pathologist were used to make a image masks, which was utilized to extract tiles from 51 WSIs, from 51 patients. Up to three magnifications of these tiles were used for model training and validation of a CNN.

Experiments were done using mono-, di- and tri-scale models which used one, two and three magnifications of tiles, respectively, for training and validation. These models where thereafter compared with each-other to find a optimal model.

The mono-scale model was shown to be the best performing model, through the use of cross validation. The final results, of using this model to predict prognosis of WSIs showed a F_1 score of 0.7667 and an AUC of 0.81.

While the dataset is obtained from a small number of patients, the evaluation of the obtained results shows, that the usage of this CNN is a promising method that is worth further exploring.

Bibliography

[1] World Health Organization The Global Cancer Observatory. *Non-melanoma skin cancer, Source: Globocan 2020*. URL: https://gco.iarc.fr/today/data/factsheets/cancers/17-Non-melanoma-skin-cancer-fact-sheet.pdf. (accessed: 17.05.2022).

[2] World Health Organization The Global Cancer Observatory. *Melanoma of skin, Source: Globocan 2020*. URL: https://gco.iarc.fr/today/data/factsheets/cancers/16-Melanoma-of-skin-fact-sheet.pdf. (accessed: 17.05.2022).

[3] Cancer Registry of Norway. *Cancer in Norway 2020 - Cancer incidence, mortality, survival and prevalence in Norway*. Oslo: Cancer Registry of Norway, 2021.

[4] Stanford University Human-Centered AI. *Machine Learning Boosts Cancer Prognosis*. URL: https://hai.stanford.edu/news/machine-learning-boosts-cancer-prognosis. (accessed: 19.05.2022).

[5] Cancer Registry of Norway. *Machine learning in cancer research*. URL: https://www.kreftregisteret.no/en/Research/about-research/machine-learning-in-cancer-research/. (accessed: 19.05.2022).

[6] Yu-Heng Lai et al. "Overall survival prediction of non-small cell lung cancer by integrating microarray and clinical data with deep learning". In: *Scientific reports* 10.1 (2020), pp. 1–11.

[7] Zhong-Si Zhang et al. "Prognostic Prediction for Non-small-Cell Lung Cancer Based on Deep Neural Network and Multimodal Data". In: *International Conference on Intelligent Computing*. Springer. 2021, pp. 549–560.

[8] Dongdong Sun, Minghui Wang, and Ao Li. "A multimodal deep neural network for human breast cancer prognosis prediction by integrating multi-dimensional data". In: *IEEE/ACM transactions on computational biology and bioinformatics* 16.3 (2018), pp. 841–850.

[9] Li-Hsin Cheng, Te-Cheng Hsu, and Che Lin. "Integrating ensemble systems biology feature selection and bimodal deep neural network for breast cancer prognosis prediction". In: *Scientific Reports* 11.1 (2021), pp. 1–10.

[10] Hani Yousef, Mandy Alhajj, and Sandeep Sharma. *Anatomy, Skin (Integument), Epidermis*. URL: `https://www-ncbi-nlm-nih-gov.ezproxy.uis.no/books/NBK470464/`. (accessed: 31.01.2022).

[11] Thomas M. Brown1 and Karthik Krishnamurthy. *Histology, Dermis*. URL: `https://www-ncbi-nlm-nih-gov.ezproxy.uis.no/books/NBK535346/`. (accessed: 31.01.2022).

[12] McGill University Department of Pathology. *What is Pathology?* URL: `https://www.mcgill.ca/pathology/about/definition`. (accessed: 08.05.2022).

[13] Mark D Zarella et al. "A practical guide to whole slide imaging: a white paper from the digital pathology association". In: *Archives of pathology & laboratory medicine* 143.2 (2019), pp. 222–234.

[14] John KC Chan. "The wonderful colors of the hematoxylin–eosin stain in diagnostic surgical pathology". In: *International journal of surgical pathology* 22.1 (2014), pp. 12–32.

[15] Mayo clinic. *Melanoma*. URL: `https://www.mayoclinic.org/diseases-conditions/melanoma/symptoms-causes/syc-20374884`. (accessed: 06.05.2022).

[16] American Cancer Society medical. *Melanoma Skin Cancer Stages*. URL: `https://www.cancer.org/cancer/melanoma-skin-cancer/detection-diagnosis-staging/melanoma-skin-cancer-stages.html`. (accessed: 31.01.2022).

[17] Wikimedia Commons. *File:Diagram showing the T stages of melanoma CRUK 373.svg*. URL: `https://commons.wikimedia.org/wiki/File:Diagram_showing_the_T_stages_of_melanoma_CRUK_373.svg`. (accessed: 05.05.2022).

[18] Jeffrey Dean. "The deep learning revolution and its implications for computer architecture and chip design". In: (2020), pp. 8–14.

[19] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[20] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable.* 2nd ed. 2022. URL: https://christophm.github.io/interpretable-ml-book.

[21] Wikimedia Commons. *File:Gradient descent.gif.* URL: https://commons.wikimedia.org/wiki/File:Gradient_descent.gif. (accessed: 16.05.2022).

[22] Rune Wetteland, Kjersti Engan, and Trygve Eftesol. "Parameterized extraction of tiles in multilevel gigapixel images". In: *2021 12th International Symposium on Image and Signal Processing and Analysis (ISPA).* IEEE. 2021, pp. 78–83.

[23] Nathan Inkawhich. *FINETUNING TORCHVISION MODELS.* URL: https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html. (accessed: 13.04.2022).

[24] Jacci Howard Bear. *The HSV Color Model in Graphic Design.* URL: https://www.lifewire.com/what-is-hsv-in-design-1078068. (accessed: 30.05.2022).

[25] Roger Amundsen. "Melanoma Diagnosis and Localization from Whole Slide Images using Convolutional Neural Networks". MA thesis. University of Stavanger, Norway, June 2022.

[26] Jason Brownlee. *Better deep learning: train faster, reduce overfitting, and make better predictions.* Machine Learning Mastery, 2018.

[27] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression.* Vol. 398. John Wiley & Sons, 2013.

Appendices

A. Overview of WSIs used in this thesis

Table A.1: Prognosis of each WSI used in this thesis. The information in this table is from the xlsx-file: *Master_god_dårlig_prognose*.

WSI name	**Prognosis**	**WSI name**	**Prognosis**
SUShud51	Bad prognosis	SUShud101	Good prognosis
SUShud57	Bad prognosis	SUShud102	Good prognosis
SUShud59	Bad prognosis	SUShud103	Good prognosis
SUShud63	Bad prognosis	SUShud104	Good prognosis
SUShud74	Bad prognosis	SUShud105	Good prognosis
SUShud76	Bad prognosis	SUShud106	Good prognosis
SUShud82	Bad prognosis	SUShud107	Good prognosis
SUShud84	Bad prognosis	SUShud108	Good prognosis
SUShud94	Bad prognosis	SUShud109	Good prognosis
SUShud120	Bad prognosis	SUShud110	Good prognosis
SUShud121	Bad prognosis	SUShud111	Good prognosis
SUShud122	Bad prognosis	SUShud112	Good prognosis
SUShud123	Bad prognosis	SUShud113	Good prognosis
SUShud124	Bad prognosis	SUShud114	Good prognosis
SUShud125	Bad prognosis	SUShud115	Good prognosis
SUShud126	Bad prognosis	SUShud116	Good prognosis
SUShud127	Bad prognosis	SUShud117	Good prognosis

Continued on next page

Table A.1: Prognosis of each WSI used in this thesis. The information in this table is from the xlsx-file: *Master_god_dårlig_prognose*. (Continued)

WSI name	**Prognosis**	**WSI name**	**Prognosis**
SUShud128	Bad prognosis	SUShud118	Good prognosis
SUShud129	Bad prognosis	SUShud119	Good prognosis
SUShud130	Bad prognosis	SUShud73	Good prognosis
SUShud131	Bad prognosis	SUShud75	Good prognosis
SUShud132	Bad prognosis	SUShud78	Good prognosis
SUShud133	Bad prognosis	SUShud85	Good prognosis
SUShud134	Bad prognosis	SUShud88	Good prognosis
SUShud135	Bad prognosis	SUShud91	Good prognosis
SUShud136	Bad prognosis		

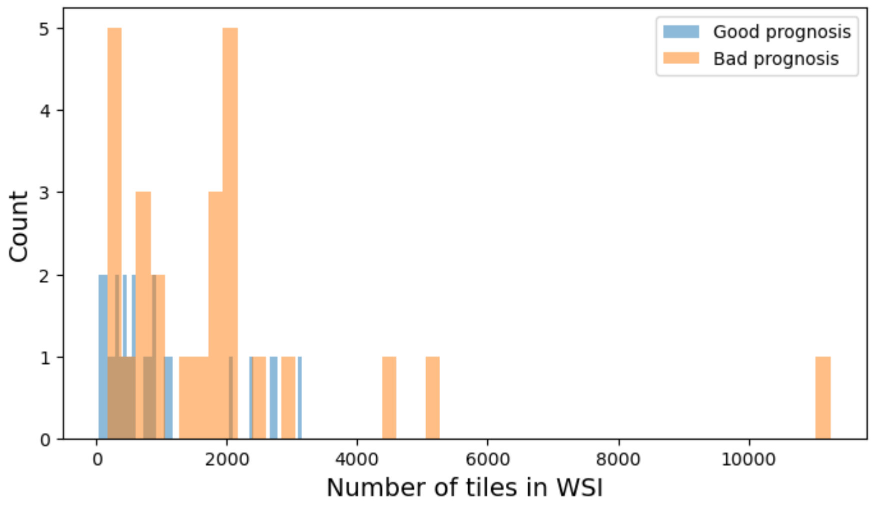


Figure A.1: Histogram over the distribution of tiles in WSIs, at 20x magnification.

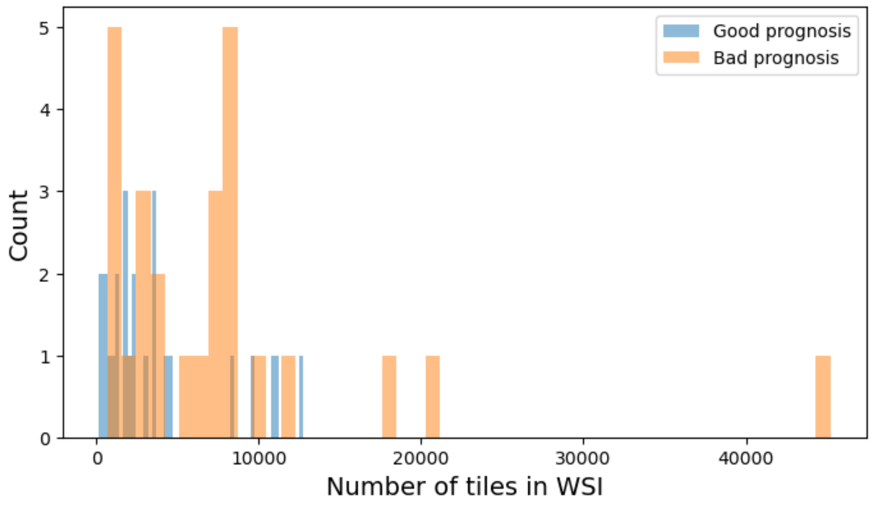


Figure A.2: Histogram over the distribution of tiles in WSIs, at 40x magnification.

B. Experiments and results details

This appendix chapter displays details from experiments and results in chapter 6. The dataset format used in this appendix is explained in section 4.1.

B.1 Details from experiment 3a

Receiver operating characteristic and prediction results

Table B.1: Data from ROC and corresponding F_1 score and accuracy. WSIs with proportion of predicted bad-prognosis tiles over threshold is predicted to have bad prognosis.

Threshold	**Sensitivity**	**1-Specificity**	**F_1 score**	**Accuracy**
0.9240	0.0385	0.0000	0.0741	0.5098
0.9040	0.1154	0.0000	0.2069	0.5490
0.9000	0.1154	0.0400	0.2000	0.5294
0.8960	0.1538	0.0400	0.2581	0.5490
0.8600	0.2308	0.0400	0.3636	0.5882
0.8017	0.3077	0.0400	0.4571	0.6275
0.7720	0.3077	0.0800	0.4444	0.6078
0.7440	0.4231	0.0800	0.5641	0.6667
0.7120	0.4231	0.1200	0.5500	0.6471
0.6920	0.4615	0.1200	0.5854	0.6667
0.6640	0.4615	0.1600	0.5714	0.6471
0.6400	0.5769	0.1600	0.6667	0.7059

Continued on next page

Table B.1: Data from ROC and corresponding F_1 score and accuracy. WSIs with proportion of predicted bad-prognosis tiles over threshold is predicted to have bad prognosis. (Continued)

Threshold	**Sensitivity**	**1-Specificity**	**F_1 score**	**Accuracy**
0.6240	0.5769	0.2000	0.6522	0.6863
0.5640	0.7308	0.2000	0.7600	0.7647
0.5520	0.7308	0.2400	0.7451	0.7451
0.5280	0.8077	0.2400	0.7925	0.7843
0.4000	0.8077	0.4400	0.7241	0.6863
0.3720	0.8846	0.4400	0.7667	0.7255
0.3380	0.8846	0.5200	0.7419	0.6863
0.3320	0.9615	0.5200	0.7812	0.7255
0.2160	0.9615	0.7200	0.7246	0.6275
0.2120	1.0000	0.7600	0.7324	0.6275
0.0764	1.0000	1.0000	0.6753	0.5098

Results from testing WSIs

WSI name	Number*	Proportion*	Predicted class	Correct class
SUShud59	215/250	0.8600	Bad prognosis	Bad prognosis
SUShud63	188/250	0.7520	Bad prognosis	Bad prognosis
SUShud76	132/250	0.5280	Bad prognosis	Bad prognosis
SUShud82	231/250	0.9240	Bad prognosis	Bad prognosis
SUShud127	186/232	0.8017	Bad prognosis	Bad prognosis
SUShud129	224/250	0.8960	Bad prognosis	Bad prognosis
SUShud78	156/250	0.6240	Bad prognosis	Good prognosis
SUShud103	225/250	0.9000	Bad prognosis	Good prognosis
SUShud105	47/250	0.1880	Good prognosis	Good prognosis
SUShud111	82/250	0.3280	Good prognosis	Good prognosis
SUShud118	53/250	0.2120	Good prognosis	Good prognosis

(a) Iteration 1.

* of tiles predicted as bad prognosis.

WSI name	Number*	Proportion*	Predicted class	Correct class
SUShud51	146/250	0.5840	Bad prognosis	Bad prognosis
SUShud84	133/250	0.5320	Bad prognosis	Bad prognosis
SUShud120	215/250	0.8600	Bad prognosis	Bad prognosis
SUShud125	162/250	0.6480	Bad prognosis	Bad prognosis
SUShud130	98/250	0.3920	Bad prognosis	Bad prognosis
SUShud73	112/250	0.4480	Bad prognosis	Good prognosis
SUShud75	18/37	0.4865	Bad prognosis	Good prognosis
SUShud102	166/250	0.6640	Bad prognosis	Good prognosis
SUShud104	85/250	0.3400	Good prognosis	Good prognosis
SUShud108	66/250	0.2640	Good prognosis	Good prognosis

(b) Iteration 2.
* of tiles predicted as bad prognosis.

WSI name	Number*	Proportion*	Predicted class	Correct class
SUShud57	226/250	0.9040	Bad prognosis	Bad prognosis
SUShud121	83/250	0.3320	Good prognosis	Bad prognosis
SUShud123	83/250	0.3320	Good prognosis	Bad prognosis
SUShud128	186/250	0.7440	Bad prognosis	Bad prognosis
SUShud136	191/250	0.7640	Bad prognosis	Bad prognosis
SUShud85	138/250	0.5520	Bad prognosis	Good prognosis
SUShud101	81/250	0.3240	Good prognosis	Good prognosis
SUShud107	24/71	0.3380	Good prognosis	Good prognosis
SUShud109	43/250	0.1720	Good prognosis	Good prognosis
SUShud112	36/250	0.1440	Good prognosis	Good prognosis

(c) Iteration 3.
* of tiles predicted as bad prognosis.

WSI name	Number*	Proportion*	Predicted class	Correct class
SUShud122	173/250	0.6920	Bad prognosis	Bad prognosis
SUShud131	141/250	0.5640	Bad prognosis	Bad prognosis
SUShud132	93/250	0.3720	Good prognosis	Bad prognosis
SUShud133	160/250	0.6400	Bad prognosis	Bad prognosis
SUShud134	226/250	0.9040	Bad prognosis	Bad prognosis
SUShud88	54/250	0.2160	Good prognosis	Good prognosis
SUShud106	73/250	0.2920	Good prognosis	Good prognosis
SUShud113	12/157	0.0764	Good prognosis	Good prognosis
SUShud115	193/250	0.7720	Bad prognosis	Good prognosis
SUShud119	21/172	0.1221	Good prognosis	Good prognosis

(d) Iteration 4.
* of tiles predicted as bad prognosis.

WSI name	Number*	Proportion*	Predicted class	Correct class
SUShud74	161/250	0.6440	Bad prognosis	Bad prognosis
SUShud94	143/250	0.5720	Bad prognosis	Bad prognosis
SUShud124	53/250	0.2120	Good prognosis	Bad prognosis
SUShud126	95/166	0.5723	Bad prognosis	Bad prognosis
SUShud135	173/210	0.8238	Bad prognosis	Bad prognosis
SUShud91	178/250	0.7120	Bad prognosis	Good prognosis
SUShud110	100/250	0.4000	Bad prognosis	Good prognosis
SUShud114	52/105	0.4952	Bad prognosis	Good prognosis
SUShud116	41/250	0.1640	Good prognosis	Good prognosis
SUShud117	119/250	0.4760	Bad prognosis	Good prognosis

(e) Iteration 5.
* of tiles predicted as bad prognosis.

Table B.2: Results from testing each WSI in validation set with cross validation. Threshold set to 0.3720.

B.2 Details from experiment 3b

Receiver operating characteristic and prediction results

Table B.3: Data from ROC and corresponding F_1 score and accuracy. WSIs with proportion of predicted bad-prognosis tiles over threshold is predicted to have bad prognosis.

Threshold	**Sensitivity**	**1-Specificity**	**F_1 score**	**Accuracy**
1.0000	0.0000	0.0400	0.0000	0.4706
0.9640	0.0769	0.0400	0.1379	0.5098
0.9600	0.1538	0.0400	0.2581	0.5490
0.9520	0.1923	0.0400	0.3125	0.5686
0.9360	0.1923	0.1600	0.2857	0.5098
0.9200	0.2692	0.1600	0.3784	0.5490
0.9160	0.2692	0.2000	0.3684	0.5294
0.8360	0.3846	0.2000	0.4878	0.5882
0.8240	0.3846	0.2400	0.4762	0.5686
0.7920	0.5385	0.2400	0.6087	0.6471
0.6440	0.5385	0.4400	0.5490	0.5490
0.5840	0.6154	0.4400	0.6038	0.5882
0.5211	0.6154	0.4800	0.5926	0.5686
0.4480	0.8077	0.4800	0.7119	0.6667
0.4280	0.8462	0.5200	0.7213	0.6667
0.4160	0.8462	0.5600	0.7097	0.6471
0.3920	0.8846	0.5600	0.7302	0.6667
0.3640	0.8846	0.6000	0.7188	0.6471
0.3440	0.9231	0.6000	0.7385	0.6667
0.2440	0.9231	0.6800	0.7164	0.6275
0.2320	0.9615	0.6800	0.7353	0.6471
0.2160	0.9615	0.7200	0.7246	0.6275

Continued on next page

Table B.3: Data from ROC and corresponding F_1 score and accuracy. WSIs with proportion of predicted bad-prognosis tiles over threshold is predicted to have bad prognosis. (Continued)

Threshold	**Sensitivity**	**1-Specificity**	**F_1 score**	**Accuracy**
0.1800	1.0000	0.7200	0.7429	0.6471
0.0058	1.0000	1.0000	0.6753	0.5098

Results from testing WSIs

WSI name	Number*	Proportion*	Predicted class	Correct class
SUShud59	230/250	0.9200	Bad prognosis	Bad prognosis
SUShud63	120/250	0.4800	Bad prognosis	Bad prognosis
SUShud76	112/250	0.4480	Good prognosis	Bad prognosis
SUShud82	240/250	0.9600	Bad prognosis	Bad prognosis
SUShud127	194/232	0.8362	Bad prognosis	Bad prognosis
SUShud129	231/250	0.9240	Bad prognosis	Bad prognosis
SUShud78	186/250	0.7440	Bad prognosis	Good prognosis
SUShud103	234/250	0.9360	Bad prognosis	Good prognosis
SUShud105	54/250	0.2160	Good prognosis	Good prognosis
SUShud111	61/250	0.2440	Good prognosis	Good prognosis
SUShud118	41/250	0.1640	Good prognosis	Good prognosis

(a) Iteration 1.
* of tiles predicted as bad prognosis.

WSI name	Number*	Proportion*	Predicted class	Correct class
SUShud51	201/250	0.8040	Bad prognosis	Bad prognosis
SUShud84	238/250	0.9520	Bad prognosis	Bad prognosis
SUShud125	218/250	0.8720	Bad prognosis	Bad prognosis
SUShud120	241/250	0.9640	Bad prognosis	Bad prognosis
SUShud130	205/250	0.8200	Bad prognosis	Bad prognosis
SUShud73	235/250	0.9400	Bad prognosis	Good prognosis
SUShud75	37/37	1.0000	Bad prognosis	Good prognosis
SUShud102	236/250	0.9440	Bad prognosis	Good prognosis
SUShud104	229/250	0.9160	Bad prognosis	Good prognosis
SUShud108	206/250	0.8240	Bad prognosis	Good prognosis

(b) Iteration 2.
* of tiles predicted as bad prognosis.

WSI name	Number*	Proportion*	Predicted class	Correct class
SUShud57	240/250	0.9600	Bad prognosis	Bad prognosis
SUShud121	107/250	0.4280	Good prognosis	Bad prognosis
SUShud123	98/250	0.3920	Good prognosis	Bad prognosis
SUShud128	204/250	0.8160	Bad prognosis	Bad prognosis
SUShud136	209/250	0.8360	Bad prognosis	Bad prognosis
SUShud85	179/250	0.7160	Bad prognosis	Good prognosis
SUShud101	104/250	0.4160	Good prognosis	Good prognosis
SUShud107	37/71	0.5211	Bad prognosis	Good prognosis
SUShud109	91/250	0.3640	Good prognosis	Good prognosis
SUShud112	36/250	0.1440	Good prognosis	Good prognosis

(c) Iteration 3.
* of tiles predicted as bad prognosis.

WSI name	Number*	Proportion*	Predicted class	Correct class
SUShud122	122/250	0.4880	Bad prognosis	Bad prognosis
SUShud131	86/250	0.3440	Good prognosis	Bad prognosis
SUShud132	45/250	0.1800	Good prognosis	Bad prognosis
SUShud133	129/250	0.5160	Bad prognosis	Bad prognosis
SUShud134	198/250	0.7920	Bad prognosis	Bad prognosis
SUShud88	22/250	0.0880	Good prognosis	Good prognosis
SUShud106	39/250	0.1560	Good prognosis	Good prognosis
SUShud113	13/157	0.0828	Good prognosis	Good prognosis
SUShud115	84/250	0.3360	Good prognosis	Good prognosis
SUShud119	1/172	0.0058	Good prognosis	Good prognosis

(d) Iteration 4.
* of tiles predicted as bad prognosis.

WSI name	Number*	Proportion*	Predicted class	Correct class
SUShud74	121/250	0.4840	Bad prognosis	Bad prognosis
SUShud94	146/250	0.5840	Bad prognosis	Bad prognosis
SUShud124	58/250	0.2320	Good prognosis	Bad prognosis
SUShud126	101/166	0.6084	Bad prognosis	Bad prognosis
SUShud135	206/210	0.9810	Bad prognosis	Bad prognosis
SUShud91	196/250	0.7840	Bad prognosis	Good prognosis
SUShud110	107/250	0.4280	Good prognosis	Good prognosis
SUShud114	70/105	0.6667	Bad prognosis	Good prognosis
SUShud116	40/250	0.1600	Good prognosis	Good prognosis
SUShud117	161/250	0.6440	Bad prognosis	Good prognosis

(e) Iteration 5.
* of tiles predicted as bad prognosis.

Table B.4: Results from testing each WSI in validation set with cross validation. Threshold set to 0.4480.