UiT THE ARCTIC UNIVERSITY OF NORWAY

Faculty of Science and Technology
Department of Mathematics and Statistics

A Step Towards Deep Learning-based CADs for Cancer Analysis in Medical Imaging

—

André Pedersen

STA-3941 Master's Thesis in Applied Physics and Mathematics, Spring 2019

Abstract

In 2018, cancer was the second leading cause of death worldwide. Early detection can reduce mortality. Screening programs intended for early detection increases the workload for clinicians. To improve efficiency CAD systems would be highly beneficial.

We have developed CAD systems using deep learning, for automatic tissue segmentation and prediction of diagnosis in lung and breast cancer. The first subproject focuses on automatic detection, 3D segmentation and malignancy prediction of lung nodules in CT, and the other aims to design an automatic method for breast tumor segmentation and histological grade prediction.

For lung nodule segmentation, we designed a new 3D-UNet architecture to handle larger input chunks than what is commonly used. Our best model achieved 0.915 recall, 2.9 FPR and 0.813 DSCTP on a subset of the LIDC data set. For malignancy prediction we designed a CNN architecture that achieved a weighted average f1-score of 0.960, only requiring a centroid initialization of the nodule.

We then designed an autoencoder for breast tumor segmentation, and achieved a DSC of 0.895 and 0.881 on two independent data sets. For histological grade prediction, we achieved a weighted average f1-score of 0.824. Using max voting we produced correct classification of 10/12 WSIs.

Acknowledgement

I would like to express my gratitude to my main advisor, Thomas Langø, for having faith in me and giving me the opportunity to work with something I truly love. Thanks for always supporting me, helping me in any way, and being available for discussions, no matter the topic.

I would also like to thank my three co-advisors Marit Valla, Hanne Sorger and Sigrunn Holbek Sørbye for their amazing support with this thesis, kind comments and many interesting discussions.

Thank you Tor-Arne Schmidt Nordmo for teaching me the ropes of machine learning and for always being available as a friend and mentor.

I also want to thank my friends at UiT for an amazing five years, especially my old class mates.

In addition, I want to thank my first grade teacher, Kari Sjøveian, for making learning fun from day one.

My greatest gratitude goes to my family for always supporting and helping me throughout my studies and in my life.

Finally, I want to dedicate my thesis to my two brothers, Tommy and Kim-Are. I hope I made you proud.

Contents

List of Figures **M**

List of Tables **O**

List of Acronyms and Abbreviations **Q**

1 Introduction **1**
1.1 Problem definition and goals 3
1.2 Summary of studies and contributions 3
1.3 Outline 5

2 Technical background **7**
2.1 Machine learning 7
2.1.1 Supervised learning 8
2.2 Artificial Neural Networks 9
2.3 Backpropagation 11
2.4 Gradient descent 12
2.5 Optimizers 13
2.5.1 Adadelta 14
2.6 Loss functions 14
2.6.1 Categorical cross-entropy 15
2.6.2 Dice loss 16
2.7 Activation functions 16
2.7.1 ReLu 17
2.7.2 Softmax 18
2.8 Training and evaluation 18
2.8.1 Overfitting problem 19
2.9 Convolutional Neural Networks 21
2.9.1 The convolution operation 21
2.9.2 Pooling 22
2.9.3 Complexity problem 23
2.9.4 Dropout 24
2.9.5 Batch Normalization (BN) 25
2.10 Classification 25

2.10.1 VGG16 26
2.11 Segmentation 27
2.11.1 UNet 28

I Lung cancer 31

3 Clinical motivation 33
3.1 Lung nodule and malignancy 35
3.2 X-ray imaging 37
3.2.1 Computed Tomography 37
3.2.2 Hounsfield Units 38

4 Lung nodule segmentation 41
4.1 Data acquisition and pre-processing 41
4.1.1 The data set 41
4.1.2 Pre-processing 43
4.2 Design 45
4.2.1 Architecture 45
4.2.2 Training 46
4.2.3 Data augmentation 47
4.3 Evaluation 47
4.4 Results 48

5 Lung segmentation 51
5.1 Data acquisition and pre-processing 52
5.2 Design 52
5.2.1 Machine learning approach 52
5.2.2 Intensity-based method 53
5.3 Evaluation 55
5.4 Results 55
5.4.1 LCTSC performance 55
5.4.2 LIDC performance 55

6 Malignancy prediction 57
6.1 Data acquisition and pre-processing 57
6.2 Design 58
6.3 Evaluation 60
6.4 Results 61

7 Lung nodule CAD performance 63
7.1 Data acquisition and pre-processing 63
7.1.1 Additional test set 63
7.1.2 Pre-processing 64

7.2 Design 64
7.2.1 Prediction thresholding 64
7.2.2 Lung mask filtering 65
7.2.3 Prediction design 65
7.2.4 Predicted size filtering 66
7.3 Evaluation 67
7.3.1 Experiments 67
7.4 Results 68
7.4.1 Detection performance 68
7.4.2 Segmentation performance 73
7.4.3 Filtering ground truth on size 74
7.4.4 Performance comparison 76
7.4.5 Integration and visualization 77

II Breast cancer **79**

8 Clinical motivation **81**
8.1 Digital pathology 82
8.2 Breast cancer diagnostics and treatment 83
8.3 Histological grade 84

9 Data acquisition and pre-processing **87**
9.1 Preprocessing 88
9.1.1 Tumor annotation extraction 88
9.2 Proposed solution 89
9.3 Tissue segmentation 90
9.4 Patch generation and data storage 91

10 Tumor segmentation **95**
10.1 Design 96
10.2 Evaluation 97
10.2.1 Monte Carlo cross-validation 97
10.2.2 Bootstrapping 98
10.2.3 Experiments 98
10.3 Results 98
10.3.1 Monte Carlo cross-validation 98
10.3.2 Bootstrapping 100

11 Patch wise classification **103**
11.1 Design 103
11.2 Evaluation 104
11.2.1 Experiments 105
11.3 Results 106

11.3.1 Multiclass classification 106
11.3.2 Binary classification 110
11.3.3 Visualization 113

III Performance analysis and discussion 115

12 Concluding remarks 117
12.1 Lung cancer 117
12.1.1 Lung segmentation 117
12.1.2 Lung nodule-CAD performance 119
12.1.3 Malignancy prediction 121
12.1.4 Correlation between FPR and resolution 122
12.1.5 LUNA design for FPR 124
12.1.6 Performance comparison 124
12.2 Breast cancer 125
12.2.1 Tissue detection 125
12.2.2 Tumor segmentation 127
12.2.3 Histological grade classification 128
12.3 Deep learning based CADs 130
12.3.1 Segmentation 130
12.3.2 Classification 130
12.3.3 Local vs open data sets 130
12.3.4 Machine learning vs traditional methods 131
12.3.5 Speed 131
12.3.6 Data augmentation 132
12.3.7 Bootstrapping 132
12.4 Conclusion 134
12.5 Future work 134

Appendices 137

A Evaluation Metrics 137

B Image Processing 141
B.1 Colorspaces 141
B.1.1 HSI 141
B.2 Thresholding 142
B.2.1 Otsu’s Method 142
B.3 Convolution 143
B.4 Mathematical Morphology 144

C Inference 145
C.1 Cross-validation 145

C.2 Bootstrapping 146
C.2.1 Accelerated bias-corrected percentile method 147

Bibliography **151**

List of Figures

1.1 Outline of thesis 6

2.1 Spectrum of AI-ML-DL 8
2.2 Multilayer neural network 9
2.3 Linear and nonlinear boundary function 10
2.4 Early stopping 19
2.5 Convolutional Neural Network 21
2.6 Pooling operators 23
2.7 Dropout 24
2.8 Macroarchitecture of VGG16 26
2.9 UNet architecture 29

3.1 Clinical stages of lung cancer 33
3.2 Flow-chart for management of SPN 34
3.3 Different lung nodule types in CT 36

4.1 3D lung nodule segmentation architecture 45
4.2 Performance of 3D-UNet 49

5.1 2D UNet architecture for lung segmentation 53
5.2 2D-UNet vs traditional method 56

6.1 CNN malignancy classifier 59

7.1 Non-overlapping predictions 65
7.2 Overlapping predictions 66
7.3 Comparison of designs in lung nodule detection performance 70
7.4 Summary detection performance 72
7.5 Segmentation performance as a function of prediction threshold 73
7.6 Filtering candidates as a function of predicted nodule size . . 74
7.7 Performance on nodule size 75
7.8 CAD prototype 78
7.9 Integration with CustusX 78

9.1 Comparison of WSI with mask 89
9.2 Tissue detector scheme 91
9.3 Image pyramid 92

10.1 2D UNet architecture for tumor segmentation 96

11.1 BC grade CNN architecture 104
11.2 Multiclass classifier 114
11.3 Binary classifier 114

12.1 Comparison between lung segmentation methods 118
12.2 Distribution of FPR 123
12.3 Tissue detection using HSV color domain 125
12.4 Tissue detection using RGB color domain 126
12.5 Segmentation performance of best network 127
12.6 Heavy HSV color augmentation 132
12.7 Comparing distributions of DSC and avDSC 133

A.1 Binary confusion matrix 138
A.2 ROC-curve 139

B.1 Morphological operators 144

C.1 K-fold cross-validation 146

List of Tables

3.1 Relation between size and malignancy of lung nodules . . . 35

5.1 Lung segmentation performance 56

6.1 Malignancy classification results 62

7.1 Seven models with different design configurations 68
7.2 Performance comparison lung nodule-CAD 76

8.1 Nottingham grading system summary 84

10.1 Performance on initial data set using monte carlo CV 100
10.2 Performance on new data set using monte carlo CV 100
10.3 Performance of final models, evaluated on initial data set . . 101
10.4 Performance of final models, evaluated on new data set . . . 101

11.1 Confusion matrices patch wise classification 107
11.2 Patch wise classification performance summary 107
11.3 Confusion matrices WSI classification 108
11.4 WSI classification performance summary 108
11.5 Confusion matrices WSI classification on new data set 109
11.6 WSI classification performance summary on new data set . . 109
11.7 Patch wise binary classification performance summary 111
11.8 Patch wise binary classification performance 112
11.9 WSI binary classification performance on new data set . . . 113

List of Acronyms and Abbreviations

ACC	Accuracy
AI	Artificial Intelligence
ANN	Artificial Neural Networks
BC	Breast Cancer
CAD	Computer Assisted Device
CE	(Categorical) Cross-Entropy
CNN	Convolutional Neural Networks
CT	Computed Tomography
CV	Cross-Validation
DL	Deep Learning
DSC	Dice Similarity Coefficient
DSCTP	Dice Score True Positive
FN	False Negative
FP	False Positive
FROC	Free-response Receiver Operating Characteristic
GGC	Ground-Glass Component
GT	Ground Truth
HE	Hematoxylin and Eosin
HU	Hounsfield Units
LP	Layer Perceptron

ML	Machine Learning
PPV	Positive Predictive Value
PR	Precision
REC	Recall
ReLu	Rectified Linear Unit
ROC	Receiving Operating Characteristic
SN	Sensitivity
SP	Specificity
SPN	Solitary Pulmonary Nodule
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
WHO	World Health Organization
WSI	Whole-Slide Imaging

/1 Introduction

According to the World Health Organization (WHO, 2018a), cancer is the second leading cause of death worldwide. Globally, one out of six deaths, or 9.6 million deaths yearly, are due to cancer alone. Early cancer detection, diagnosis and treatment can reduce cancer mortality by one third, and are therefore main priorities in the WHO cancer control strategy (WHO, 2018b).

The most common cancer types are lung, breast, colorectal, prostate, skin and stomach cancer, respectively. Lung cancer and breast cancer account for 2.09 million cases each. Most cancer-related deaths are caused by lung cancer, while breast cancer comes in fifth, with 1.76 and 0.63 million deaths per year (WHO, 2018a).

Suspected cancer is often detected by medical imaging, which directs the diagnostics strategy and clinical decision-making. A definite diagnosis also requires a biopsy with subsequent tissue analysis by a pathologist. Increasing cancer incidence, extensive use of medical imaging, and the increasing number of biopsies taken each year, represent major challenges for the health care system. Hence, the capacity of expert diagnosticians, such as radiologists and pathologists, are pushed to the limit. The increasing workload may lead to a higher risk of human error, misdiagnosis and assignment of suboptimal treatment protocols. This further affects patient quality of life and increased costs for healthcare systems (Mossel, 2018).

Cancer screening programs add to the number of medical images and tissue samples taken each year. In Norway, screening programs are already implemented for breast cancer, colon cancer and cervix cancer. Implementation of a lung cancer screening program in Norway is also debated, due to favorable data in recent international screening trials (Han *et al.*, 2018). Such screening will further add to the workload of radiologists, pulmonologists, and pathologists.

The diagnosis an expert make is also dependent on human factors. This is especially clear in pathology, where the inter-observer variability between pathologists is quite high (van Dooijeweert *et al.*, 2019, Robbins *et al.*, 1995). Even the same expert diagnostic conclusion might differ, due to the heterogeneity and complexity of histopathological images, as well as human factors.

In radiology, detection of lung nodules that may represent early stage lung cancer is typically a tedious and challenging task, because of their small size and heterogeneity. To improve the efficiency and consistency of cancer diagnostics, a computer aided design system (CADs) would be highly beneficial.

CAD-systems have been an active research field for decades, but not until recently have CAD performance in cancer diagnostics been comparable to human experts. This is mainly due to three factors: 1) The introduction of CNNs and improvements in deep learning; 2) Increase in computing power and access to GPU processors; and 3) Availability of large amount of annotated data sets, such as *LIDC-IDRI* (Armato III *et al.*, 2011) and *BACH* (Aresta *et al.*, 2018).

Convolutional Neural Networks (CNNs) have proven brilliant in image analysis, and have completely outperformed traditional state-of-the-art methods for most image analysis tasks (Razzak *et al.*, 2017). The main concept is to let the network find which features are relevant from the image to solve a specific task, which differ from traditional methods where these features were user-defined. Studies show that CNNs do a better job in selecting features to solve a task compared to humans, especially for more challenging problems. Due to the easy nature of the approach, it can be customized to suit "any" problem, given sufficiently, suitable annotated data. This versatility has led to the growing popularity of CNN-based CAD-systems.

Aiming to assist both radiologists and pathologists, this thesis describes how deep learning can be used to develop CADs to solve some common diagnostic tasks. Due to lack of accessibility of patient characteristics in data sets, the golden standard is set by expert annotators. Therefore, it was beyond the scope of this thesis to create CADs that could outperform the experts, but rather make tools able to assist them in the future.

The first subproject of this thesis, studies whether deep learning can be used for lung nodule detection and diagnostics from CT images - assisting radiologists. Next, the thesis explores whether similar techniques can be applied for breast cancer diagnostics using digital whole-slide images (WSI) - assisting pathologists. The two cancer types were chosen partly based on data accessibility, but also because they are the two most common cancers worldwide, facing similar problems in the diagnostic work-up. It also makes us able to study how the same state-of-the-art machine learning methods can be used on several different applications - across widely different imaging modalities.

In addition, the fields of cancer diagnostics in breast and lung are two of the main areas of research in the cross-disciplinary team at SINTEF (Medical Technology), St. Olavs hospital and NTNU, hosting this master's thesis work.

1.1 Problem definition and goals

Overall, the main objective of this thesis is:

To study whether one can develop state-of-the-art CADs using deep learning, which should be able to process from the raw data, and solve the task "in real time". The output should also be represented in a way that the clinicians find useful.

We will be working with two different problems, for two different cancer types, for two widely different imaging modalities. Thus, it is natural to split the problem further into two objectives:

- Design an automatic method for lung nodule detection, 3D segmentation and malignancy prediction for thoracic Computed Tomography (CT)

- Design an automatic method for breast tumor segmentation and histological grade prediction for Whole-Slide Imaging (WSI)

1.2 Summary of studies and contributions

Studies and contributions are split into three parts, as we first study each cancer type individually, and then attempt to evaluate the use of deep learning-based CADs in cancer diagnostics. The main studies and contributions for each of the three parts are listed below. Novel and unique contributions from this master's thesis work are in *cursive*.

Lung cancer diagnostics:

1. *Designed and evaluated a new 3D-UNet architecture, using much larger input chunks compared to other published work in the field. Showed that more global information during training may provide better generalization than current state-of-the-art approaches*

2. *Thorough study to find which post-processing techniques of trained models might be most beneficial in detection and 3D segmentation of lung nodules*

3. Designed a CAD system for automatic lung nodule detection, 3D segmentation and malignancy prediction from CT-images, which processes from the raw DICOM format close to "real time"

4. Evaluated and compared 2D-UNet and traditional method for lung organ segmentation for best integration with CAD

5. Studied different design choices in creating a malignancy classifier for best integration with CAD

6. Produced a prototype of the CAD, which makes the user able to view generated candidates both in 2D and 3D, as well as remove or add nodules - can be run on both CPU and GPU

7. Showed that the produced predictions can be easily integrated with the SINTEF developed software CustusX (Askeland *et al.*, 2015)

Breast cancer diagnostics:

1. *Evaluated the performance of a CNN-based patch wise classifier for histological grade (I-III) prediction from gigapixel resolution whole-slide images (WSIs), based on local information*

2. *Designed a CAD system for processing WSI from the raw cellsens vsi format to give histological grade predictions and produce confidence heatmaps between and within grades, which processes close to "real time*"

3. Designed a tissue detector

4. Trained and evaluated a 2D-UNet architecture for automatic breast tumor segmentation

5. Evaluated the use of HSV color augmentation in breast tumor segmentation and histological grade prediction in WSI

CAD systems in cancer analysis:

1. *Designed a multipotent pipeline for deep learning-based CAD systems for cancer analysis, and showed that it can be easily adapted between cancer types and imaging modalities, processing close to "real time" in all cases studied*

2. *Evaluated state-of-the-art deep learning-based methods for segmentation and classification, in terms of different imaging modalities and data types - 2D, 2.5D, 3D, gigapixel resolution 2D.*

3. Compared traditional methods against deep learning methods in terms of segmentation for insufficient data sets, for two widely different cancer types, imaging modalities and data types

1.3 Outline

In chapter 2, machine learning and neural networks are introduced, from fundamental theory to image segmentation using CNNs.

Then, we found it necessary to split the thesis into two parts, as each subproject was independent of the other. Part 1 concerns lung cancer diagnostics, chapter 3-7. Part 2 is focused on breast cancer diagnostics, chapter 8-11. Part 3 contains a performance analysis and discussion of each problem, as well as discussion of the use of deep learning in the development of CAD systems for cancer analysis. Finally, there is conclusion and a future work section. An illustration of the outline of this thesis is given in Figure 1.1.

We chose to include an appendix. These are additional concepts which might be seen as preliminaries for working with machine learning (on images). However, since we are using such a wide variety of concepts across fields, it is important to have some fundamental understanding in these fields as well, i.e. image processing and inference. Thus, it is natural to separate these from the machine learning background given in chapter 2.

From Figure 1.1, (1) shows the overall structure of the thesis. (2) shows how each subproject part is structured. For each subproject, there are multiple independent studies, which we in the end will merge to produce a final CAD system. (3) shows how each study in each subproject is organized. All blue

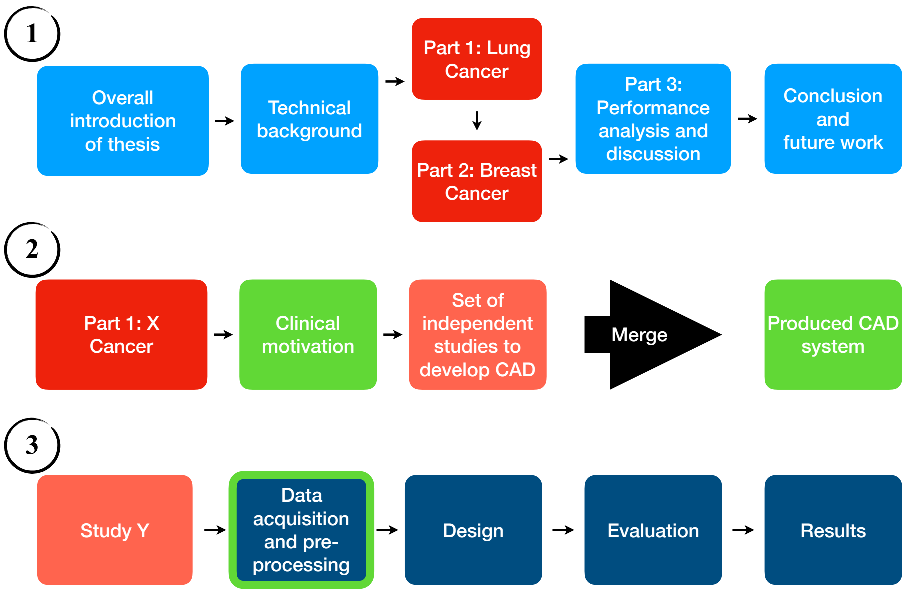

1
Overall introduction of thesis
Technical background
Part 1: Lung Cancer
Part 2: Breast Cancer
Part 3: Performance analysis and discussion
Conclusion and future work
2
Part 1: X Cancer
Clinical motivation
Set of independent studies to develop CAD
Merge
Produced CAD system
3
Study Y
Data acquisition and pre-processing
Design
Evaluation
Results

Figure 1.1: Outline of the thesis

is true for all lung cancer studied. There is a green boundary around the first box. That is because for breast cancer, there is only a single, common data acquisition and pre-processing, and thus belongs prior to Study Y.

/2 Technical background

In this thesis, we will be studying deep learning-based methods, and how they can be used to develop CAD systems. In order to design such systems using deep learning, it is necessary to have a solid understanding of machine learning, as well as different learning concepts. Therefore, in this chapter, we will start by introducing basic machine learning theory. By the end of the technical background, the necessary parts for using deep learning to solve problems, like classification and segmentation of images, will have been introduced. Note that it is recommended to also read the appendix before moving on to chapter 3, as it introduces some non-machine learning concepts; evaluation metrics, image processing and inference.

2.1 Machine learning

Computers are based on elementary arithmetic operations and gates, and need explicit instructions of how to use these to solve a specific problem. Daily tasks easily solved by humans, like detecting cars and faces, are extremely difficult for machines. A way of using a machine to solve such problems, could be by defining an algorithm, or providing a set of instructions, telling the machine how to process the input data to give an output. As problems become more complex, it becomes more challenging to make such algorithms. This blossomed the idea to make machines that could **learn** how to solve the problem, instead of being given explicit instructions. This is what is called **Machine Learning**

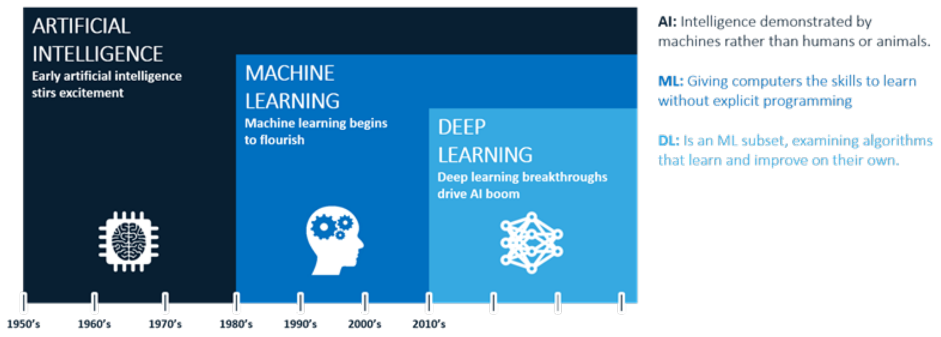

ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement
MACHINE LEARNING
Machine learning begins to flourish
DEEP LEARNING
Deep learning breakthroughs drive AI boom
AI: Intelligence demonstrated by machines rather than humans or animals.
ML: Giving computers the skills to learn without explicit programming
DL: Is an ML subset, examining algorithms that learn and improve on their own.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

Figure 2.1: Spectrum of AI-ML-DL and how they relate (Latinovic, 2018).

(ML). The idea is based on how humans learn. In order be able to separate dogs and cats, we will have to see a lot of example data and be told which objects are cats and which are dogs. By doing this many times, we would eventually learn specific cat and dog features, which makes us able to distinguish between them afterwards.

From introducing this learning concept and problem-solving ability, machines are given the ability to mimic human cognitive functions. This is what is called **Artificial Intelligence** (AI). ML is a subgenre of AI, only involving concepts based on learning. A subgenre of machine learning is **Deep Learning** (DL). It is based on the concept that data have some kind of hierarchy of complexity. By learning increasingly more complex features from the input data, it becomes possible for the machine to solve more complex problems.

2.1.1 Supervised learning

There are three main branches of learning: supervised, unsupervised and semi-supervised. In supervised learning, there exists a ground truth (GT). Labels, or ground truth, are typically denoted $\boldsymbol{y}$. In the two-class case there are two different arbitrary values, i.e. $\boldsymbol{y} \in \{0, 1\}$. In unsupervised learning, networks are trained without a ground truth. Semi-supervised learning is a mix between these two, and learning is based on both labelled and unlabelled data, but typically mostly unlabelled data.

If sufficient annotated data is available, supervised learning would theoretically always provide better results than unsupervised learning. However, supervised learning requires a lot of annotated data in order to generalize well. For that reason, unsupervised approaches might be more suitable in some cases, i.e. image segmentation. For many applications, clustering based on only pixel intensities is sufficient to segment the object(s) of interest, but as problems become more complex, supervised learning is necessary to achieve better performance.

2.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) is a framework for machine learning algorithms to process complex input data $\boldsymbol{x}$.

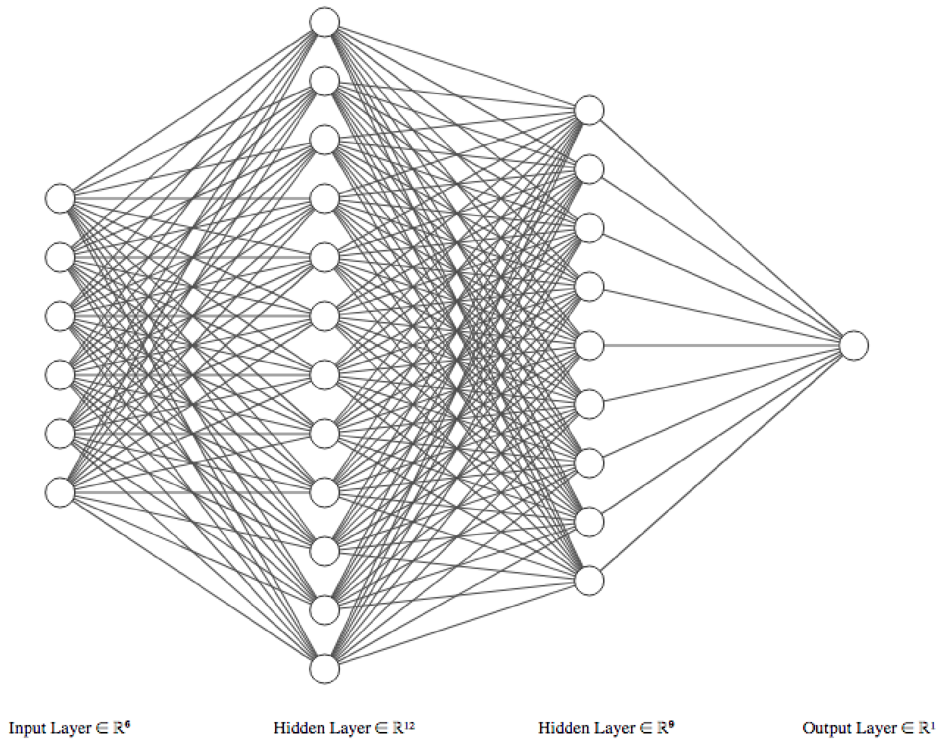

Input Layer $\in \mathbb{R}^6$
Hidden Layer $\in \mathbb{R}^{12}$
Hidden Layer $\in \mathbb{R}^9$
Output Layer $\in \mathbb{R}^1$

Figure 2.2: An illustration of how a multilayer neural network might look. In this case we have a three-layer perceptron. Figure was made using NN-SVG (LeNail, 2019).

The concept of ANN is based on what we think is the biological nature of the brain. When humans make a conclusion, they base it on a lot of different factors/inputs, which they **weight** in some manner to come to a final conclusion. Each of these inputs are called **neurons**, which have a similar interpretation as in neuroscience. The most fundamental neural network, based on a single **layer**, is called the **perceptron**, or single-layer perceptron (1LP).

By layer we mean a set of neurons which receive different weighted outputs from the same previous set of neurons. One example of a layer is the input layer, which contains the input data to be processed.

One can think of a single-layer perceptron as a dot product between the input $\boldsymbol{x}$ and the set of weights $\boldsymbol{\omega}$, also introducing some bias term ω_0, which yield:

$$y = \boldsymbol{\omega}^T \boldsymbol{x} + \omega_0 = \sum_i \omega_i x_i + \omega_0 \tag{2.1}$$

This is a linear transform. Thus, we say that the single-layer perceptron is a **linear classifier**. It receives some input data $\boldsymbol{x}$, and by a linear transform classifies the output to some value y. If the ideal output is in a specific range, i.e. $y \in [0, 1]$, it might be of interest to use some **activation function** f, which yields:

$$y = f(\boldsymbol{\omega}^T \boldsymbol{x} + \omega_0) = f(g(\boldsymbol{x})) \tag{2.2}$$

For any nonlinear activation function, the resulting classifier is also nonlinear.

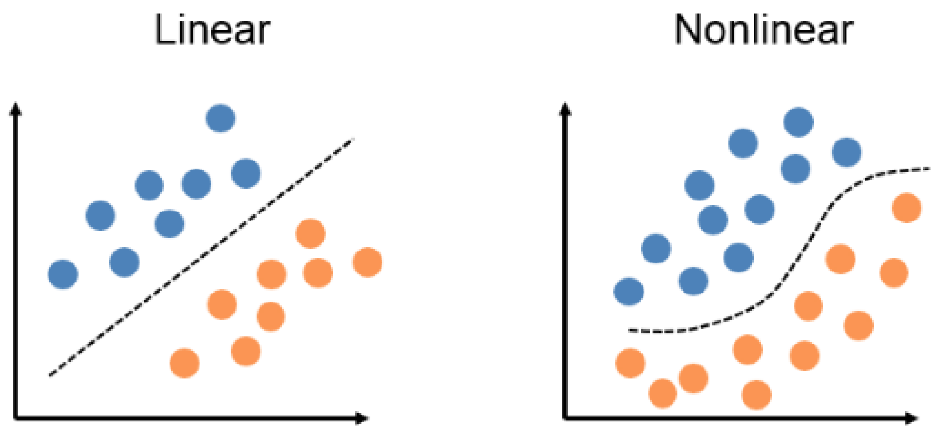


Figure 2.3: Difference between linear and nonlinear boundary function (Sullivan, 2017).

When we use a linear classifier, we assume that it is possible to separate the classes in the data by a linear boundary function g. For real-world applications, a nonlinear classifier is often preferred since it might generalize better and handle more complex problems.

For more complex problems, it may be necessary to use more layers to achieve more complex boundary functions. These new layers are referred to as **hidden layers**. By introducing more layers, more advanced mappings can be made. Data is transformed, making it more linearly separable in the transformed domain, while in the original domain it looks like some complex boundary function.

The resulting set of layers of neurons is what we refer to as an **architecture**. It defines how the data is processed in order to provide the final output.

If we were to classify images of cats and dogs, a natural metric for evaluation would be classification accuracy, i.e. how many cats and dogs are classified correctly. But how do we actually maximize classification accuracy? For a specified network, this is done by finding the optimal set of weights. They are chosen by a method called **backpropagation**, which is how *the network learns* how to set the weights to get the optimal output.

In order to use backpropagation, a cost-function or **loss function** J is needed. Its purpose is to control how the weights are set. Depending on the choice of loss function, the loss function is to be minimized/maximized. An example of a loss function is the Sum of Squared Error (SSE) loss, defined as:

$$J_{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.3}$$

where y_i denotes the ground truth, and $\hat{y}_i$ is the predicted class, for n samples.

If SSE is low, it indicated that the overall predictions $\hat{\boldsymbol{y}}$ and ground truths $\boldsymbol{y}$ are similar, reflecting a desired result. Hence, this loss function can be used for classification. The accuracy metric (Appendix A) is not differentiable. That is why we use the loss function, instead of the accuracy directly. In order to set the weights to minimize the loss function, an updating scheme called **gradient descent** is commonly used. **Backpropagation** is used to calculate the gradients. To calculate the gradients, the loss functions has to be differentiable. In the following sections, a more elaborate explanation of the terminology for ANN will be provided.

2.3 Backpropagation

When performing optimization, we use gradient descent to explore the parameter space to find the optimal solution. Backpropagation helps us to find the gradients which enables gradient descent. A full derivation of backpropagation is beyond the scope of this thesis. Thus, we will take a high-level approach to explain the method in more detail.

First, all weights are randomly initialized by a pre-defined pseudorandom sequence generator. The input is then propagated forward through the network. This is called a **forward pass**. As it propagates forward, calculations at each layer are made. At the output layer, it outputs a prediction, and an error is reported. Backpropagation then takes place, and gradients and respective errors are calculated at each layer.

Based on these estimations, the weights are updated. This process can be summarized as such:

$$\omega_j^r(new) = \omega_k^r(old) + \Delta\omega_j^r \tag{2.4}$$

$$\Delta\omega_j^r = -\mu \sum_{i=1}^{N} \delta_j^r(i)\boldsymbol{y}^{r-1}(i) \tag{2.5}$$

where $\Delta\omega_j^r$ corresponds to the update of weights at step j in layer r, for a neuron k. δ_j^r corresponds to the actual gradients at step j in layer r, while μ is called the learning rate - a learning-parameter to be explained in section 2.4.

The idea is to propagate N samples through the network, calculate all weights and errors, and at the end update all weights based on the total contribution from all samples. This is important, because if we update the weights for each sample, it would be highly random and sensitive to outliers. Therefore, for a more stable updating scheme, it is natural to base it on more samples.

How the gradients are actually calculated is based on the chain rule, and it requires quite a lot of derivations in order to explain in detail. Thus we will not give further information on the topic. A deeper understanding of backpropagation can be found in Theodoridis and Koutroumbad (2009).

Hence, when we introduce updating of weights, we define an updating scheme based on gradients found from backpropagation. The scheme introduced in section 2.4-2.5 is commonly referred to as gradient descent.

2.4 Gradient descent

The most popular way of **optimizing** neural networks is by gradient descent (Ruder, 2016), which is the concept typically used in backpropagation. By optimizing a neural network, we mean how the network iteratively searches for a lower minimum (or higher maximum) of the loss function. If a greedy approach was done (only accept solution if lower minimum for each iteration), the optimization could get stuck in local minima. Therefore, it is necessary to use a more advanced approach.

Gradient descent is a way of minimizing some arbitrary (loss) function $J(\boldsymbol{\theta})$ that is parametrized by the model $\boldsymbol{\theta} \in \mathbb{R}^d$, where d denotes the dimension. It is done by updating the parameters $\boldsymbol{\theta}$ in the *opposite* direction of the gradient of the loss function with respect to $\boldsymbol{\theta}$.

During this descent, it is necessary to introduce a learning rate μ, which controls the number of steps taken for the function to reach a (local) minimum. Another way of understanding it, is by how much weight there should be on new information. Using a too small value for μ, training might be slow and it might get stuck in a local minima. By using a larger μ, it might jump across these smaller local valleys, but might overshoot the optimum.

As discussed earlier, there is a strong link between gradient descent and back-propagation, and the resulting updating scheme can be summarized mathematically as (Theodoridis and Koutroumbad, 2009):

$$\omega_j^r(i+1) = \omega_j^r(i) - \mu \nabla_\omega J(\omega) = \omega_j^r(i) - \mu \delta_j^r(i) \boldsymbol{y}^{r-1}(i) \tag{2.6}$$

The updating procedure happens more than once. The training data is fed into the network in an iterative fashion, until the algorithm converges, or it has reached some stopping criteria. Each successive forward-backward propagation of all the training data is called an **epoch**.

It might be beneficial to update more often as a regularization technique. Therefore, it is common to split the data into smaller **chunks** and update the model after these chunks. This might speed up training, as well as making it able to find a better minima.

2.5 Optimizers

The updating procedure explained above, is called the **vanilla batch gradient descent method**. It has some limitations. Although it is simple and intuitive, it has problems exploring the full parameter space. Especially in the cases where the gradients are small, e.g. close to saddle points.

Thus, it is necessary to introduce the concept of momentum ρ, which can be used to accelerate the process in slow regions. By tuning it properly, it is also possible to make it jump away from local minima by pushing it all the way to a new hill. It easily fits into the ordinary gradient descent scheme by simply multiplying the old weights by ρ. This results in a new optimizer commonly called **stochastic gradient descent**(SGD) with momentum:

$$\omega_j^r(i+1) = \rho \omega_j^r(i) - \mu \nabla_\omega J(\omega) \tag{2.7}$$

Typical values of ρ are 0.9 or higher (Ruder, 2016). This optimizer uses samples that are drawn randomly (or shuffled) during training for each epoch, thus it is *stochastic*.

This is still one of the most popular optimizers. However, the method has limitations, e.g. trouble *navigating ravines*, which results in convergence being quite slow. By ravines we means areas where the surface curves much more steeply in one dimension than in another (Sutton, 1986). It is also extremely sensitive to hyperparameters. Hence, performing a **grid search** might be necessary, which involves a (*time consuming*) systematic search to find which set of hyperparameters produce the optimal model.

In many cases, a grid search is not feasible. In 3D lung nodule segmentation, which typically involve using highly computationally expensive methods, training time on ***one*** set of hyperparameters might take weeks. Therefore, it is necessary to study more adaptive learning schemes.

2.5.1 Adadelta

Without going too much into details, the main idea of this optimizer is to have a more constrained, monotonically decreasing learning rate, *by restricting the window of accumulated past gradients to some fixed size* (Ruder, 2016).

This is achieved by studying the root mean squared error of parameter updates, which is defined as:

$$RMSE(\nabla\omega)_{i-1} = \sqrt{E[\nabla\omega^2]_i + \varepsilon} \tag{2.8}$$

where ε is a small arbitrary constant to avoid dividing by zero in equation (2.9). The RMSE is approximated using the past parameter updates. Using this concept, it can be shown that we get the Adadelta update scheme defined as:

$$\omega_i = \omega_{i-1} + \nabla\omega_i = \omega_{i-1} - \frac{RMSE[\nabla\omega]_{i-1}}{RMSE[g_i]} g_i \tag{2.9}$$

where g_i denote that gradient of the cost function at time i.

Note that Adadelta does not explicitly use momentum, as with momentum SGD. Therefore, a natural extension to Adadelta is **Adam**, which effectively is a first-order smoothing on the gradients, which empirically have shown to give faster convergence (Ruder, 2016). However, Adam was not used in this thesis. Thus, we will not explain it any further.

2.6 Loss functions

Choosing the right loss function is key in the training phase of neural networks, and it is necessary in order to achieve optimal output. The choice of loss function depends on the task. As discussed in section 2.2, for classification of dogs and cats, *SSE* might be a good choice. Even though it is intuitive, it has its limitations, i.e. problems with smaller gradients.

Another issue in training is that the different classes might be unequally represented, i.e. many more images of ones (cats) than the other (dogs). If we used simple *SSE* as the loss function and were to train a dog/cat classifier based on this data set, the network would most likely only end up guessing only cat. The reason is that guessing cat every time generates less errors than guessing dog. Thus, it is penalized too much on the dog class, or too little on the cat class.

This problem of **unbalanced data sets** should be handled before doing any training. It can either be done by resampling the data such that they are balanced during training or using a loss function which weights penalizes classes properly during training.

In segmentation, unbalanced data sets become more challenging, as it is difficult to physically balance the classes. In this case, it may be necessary to introduce a loss function to handle the unbalanced data sets *during* training.

2.6.1 Categorical cross-entropy

One of the most commonly used loss functions in classification is called **Categorical cross-entropy** (CCE, or simply CEs). The idea is to minimize the entropy of each class. In other words, what defines each class is minimized, such that they become distinguishable.

For M number of classes, CE can be defined as:

$$CE = -\frac{1}{M}\sum_{i=1}^{M} g_{0i} log(p_{0i}) \tag{2.10}$$

where p_{0i} and g_{0i} correspond to predicted probability of each class and ground truth, respectively, where class i is the foreground class.

With this notation, it is natural to introduce the concept of **one-hot encoding**. If you have M classes, and you want to say that a sample belongs to class m, you make a binary vector which is only high for the class it belongs to. Note that this is also an assumption with CE - there are only hard memberships.

2.6.2 Dice loss

CE is quite general. It can be adapted easily to any classification problem, and there are loss functions that are specialized for more specific tasks. One such loss function is **Dice loss**, which is specifically designed for segmentation. One of the most popular metrics to evaluate segmentation is **dice score** (DSCs), which is defined as the overlap between the prediction and ground truth array for the class(es) of interest. The higher the overlap, the better the segmentation. The mathematical definition can be seen in appendix (A.5).

The idea is to design a loss function that maximizes DSC, as the loss function is minimized. The simplest way to accomplish this, is to make a binary inverse transform, which is a common way of turning a minimization problem to a maximization problem, or *vice versa*, in optimization.

For a two-class case, dice loss can be defined as:

$$DL = 1 - DSC = 1 - \frac{2|G \cap P|}{|G| \cup |P|} \tag{2.11}$$

where G and P denote the ground truth and prediction volume. Note that the denominator may hit zero if only one class is represented. Therefore, a smoothing coefficient v is usually included. This modification results in the **soft dice loss** (SDL):

$$SDL = 1 - \frac{2|G \cap P| + v}{|G| \cup |P| + v} = 1 - \frac{2\sum_{i=1}^{N} p_i g_i + v}{\sum_{i=1}^{N} p_i^2 + \sum_{i=1}^{N} g_i^2 + v} \tag{2.12}$$

Ideally the smoothing coefficient should be as small as possible, for the relation between DL and DSC to be as linear as possible. A typical value of the smoothing coefficient is 1, since it is the smallest possible overlap one can get. This introduces a smaller error in the estimations, but it is necessary in order to use this loss. Note that SDL is not differentiable as is the case with *SSE* and *CE*. This is a drawback with the loss function, since it results in problems during optimization.

2.7 Activation functions

Another crucial component in any neural network is **activation functions**. The main goal of these is to introduce nonlinear transformations between layers, such that nonlinearly separable data might still be separable by a classifier.

Without these a neural network will not be able to learn or model more complicated features within the data. This is essential in deep learning. Using these functions, it is possible to transform data to a domain of interest, i.e. force output values to be in the range $[0, 1]$, to reflect probabilities.

2.7.1 ReLu

Arguably the most used activation function in deep learning is Rectified Linear Unit ReLu. It is defined as:

$$R(x) = \begin{cases} x & , x > 0 \\ 0 & , x \leq 0 \end{cases} \tag{2.13}$$

because of it is simplicity and convergence speed compared to the more traditional activation functions, like *Sigmoid* and *Tanh* (Nwankpa *et al.*, 2018).

A problem with backpropagation, is that gradients tend to get quite small. This results in weights not being updated, and the neural network may not progress. This is called the *vanishing gradient problem*, and most traditional activation functions do not handle this problem. By introducing rectifiers, as in ReLu, gradients are forced to saturate only in one direction, which makes them less prone to vanishing gradients, since they do not get stuck as easily.

The main downside with ReLu, is that it is less general than Sigmoid and Tanh. ReLu can only be used in hidden layers, and therefore you still need the traditional ones in the output layer to make predictions.

Another well-known problem with ReLu is that gradients may vanish/die during training, meaning that a neuron might never be activated again and neglected during further training. This is referred to as the *dead neuron problem*, and it can be solved by introducing a modification to the original ReLu.

The idea is that instead of setting it to zero for values of x lower or equal to 0, you include a small gradient value. This way the gradients would never be exactly zero, and dead neurons do not occur. Because of this behaviour this new modified ReLu has been given the name **Leaky ReLu**. Mathematically it can be described as:

$$f(x) = \begin{cases} x & , x > 0 \\ \alpha x & , x \leq 0 \end{cases} \tag{2.14}$$

where α corresponds to the small gradient chosen for values less than 0. Typically, $\alpha = 0.01$ is used.

2.7.2 Softmax

In the last layer, the sigmoid or **softmax** activation function is commonly used. The difference between these two is that softmax works for any number of classes, while sigmoid only works in the two-class case. One can view softmax as a special case of sigmoid, and therefore softmax is commonly the preferred choice as long as the input is structured properly, i.e. using one-hot encoding.

For K classes, the softmax function can be defined as:

$$\sigma(\mathrm{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ }, j = 1, \ldots, K. \tag{2.15}$$

Softmax can be understood as a generalization of the logistic function - forcing values to the range [0, 1]. In the prediction of an input, each class is assigned values that sum to 1. Hence, softmax acts like a confidence predictor. That way the most probable class can be found from extracting the class with the highest softmax output, which would be the class the network has the highest confidence that the sample belongs to.

The output may be interpreted as probabilities. However, it only bases its confidence on what it has seen during training, and in the output layer it is *forced* to make a prediction. Results should be interpreted with caution, and if an outlier input, differing from the ones seen in training is given, its behaviour cannot be explained or controlled. If one trained a dog/cat-classifier, and the input was a carrot, it would still classify the carrot as either a cat or a dog.

Therefore, for instance in the two-class case, setting a threshold at 0.5 might not always be the best choice, as we cannot directly always interpret it as 50 % probability.

2.8 Training and evaluation

During training of a classifier, it is common practice to split the data set in two: **training set** and **test set**. The classifier is based on the training data. To properly evaluate the model, we need to use a data set the classifier has not seen before, i.e. test set.

It can be problematic if the test set is different (or simpler), than the training set. An example would be if we handpicked simple cases for the test set. The resulting classifier would naturally perform better on the test set, and hence we have introduced a bias. Therefore, allocation of training data is best achieved by random sampling, but this depends on how the data set is structured.

Still after this split, we have only a single estimate of the performance based on the test set. To get a better estimate it is common to do **K-fold cross-validation**, further explained in appendix (C.1). The main idea is to get a better estimate of the performance, by evaluating on effectively a "larger" data set. Cross-validation is often done when the data set is small. For larger data sets, there should not be any significant difference in the inference one makes from a single split or multiple.

2.8.1 Overfitting problem

For a given training set, we could keep training a network until we obtain zero loss. This means that the model would have fitted the training data perfectly. However, if the same model is applied on new data, the performance will typically be worse. That is because the network has been overfitted towards the training set, and we refer to this as **overfitting**.

Therefore, in order to assess whether the network has overfitted during training of a neural network, it is necessary to study its performance on an independent data set *during training*. If we used the test set for this assessment, we would have introduced a bias, since the test set would not be completely independent from training. Hence, it is common to split the data set in *three*, by also introducing a **validation set**.

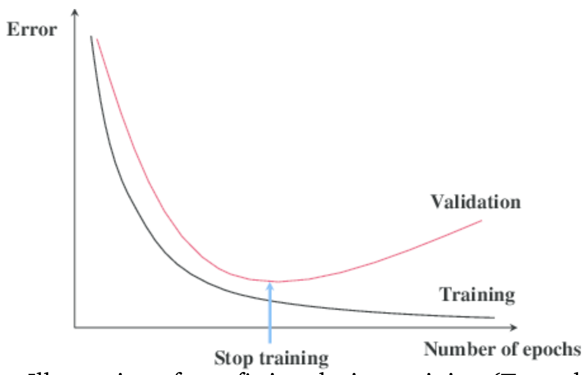


Figure 2.4: Illustration of overfitting during training (Tretyakov, 2017).

In the case of overfitting, there will be an increase in validation loss, as the training loss is decreasing. If this happens, the network has started to overfit, and all new updates would result in a poorer generalization. Note that during training, the loss function might oscillate, and therefore even though validation loss has increased after an epoch, it does not necessarily mean that overfitting has occurred. Therefore, it is necessary to study the trend over time. If for m epochs, the validation still has not decreased, it has most likely started to overfit, *or it has converged*. This would be a good place to stop, illustrating the concept of **early stopping**, as seen in Figure 2.4.

Another solution would be to save the model with lowest validation error. The problem then is that there is no stopping criterion and optimizing training parameters might also be less efficient.

If overfitting occurs too early, it is most likely because the training set is too small, the network chosen is too complex, or that you are not *regularizing* your network properly. In general, there are four general approaches to handle overfitting:

1. Increase training set
2. Reduce network size
3. Use regularization
4. Artificially expand training data

For many applications, increasing the training set is not feasible, but if it is, it most likely improves performance and increase generalization. Other solutions could be to reduce network size, or experiment with hyper-parameters. By decreasing network size, the total number of parameters will decrease, and the network will not overfit as easily.

In deep learning, L^2-regularization, dropout and batch normalization are typically used. The overall idea is to make the network able to learn smaller weights while keeping the other parameters constant.

If more data is not available, a common technique, especially for images, is to expand the training data by generating artificial images by transforming samples from the original data set. This is called **data augmentation**. These regularization techniques will be further covered in section 2.9.

It should also be mentioned that there is the possibility that using a smaller network to avoid the risk of overfitting, results in the network **underfitting**. This can be understood as the network not learning complex enough features to solve a task, and effectively performs worse overall - not just for the test set.

2.9 Convolutional Neural Networks

Convolutional Neural Networks(CNNs) represent a more complex approach in particular for image recognition tasks, i.e. classification and segmentation.

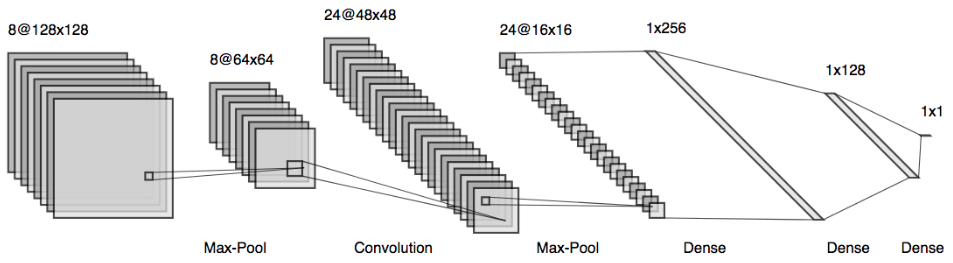


Figure 2.5: Illustration of how a Convolutional Neural Network might look like. Figure was made using NN-SVG (LeNail, 2019).

Working with images, it can be challenging to determine how to use the input data optimally to solve a task. Traditionally, user-defined features have been extracted from an image, and a network is trained based on these to solve a task.

It is challenging to handcraft features - especially as problems become more complex. Therefore, general feature extractors like HOG, SIFT, SURF and colornames have been made. They tend to be suitable for many computer vision tasks. Unfortunately, for more complex tasks, these generic feature generators fail to produce satisfactory classifier accuracy (Guérin *et al.*, 2017, Fischer *et al.*, 2014, Loussaief and Abdelkrim, 2018).

Therefore, instead of using handcrafted features to solve a task, we could train a network to find which features that are relevant to solve the task. This is the idea of CNNs. The overall concept is to generate features using the convolution operator, and then learning which features are relevant for a specific task by supervised learning using a neural network. Using CNNs, we can also easily extract information from different magnification levels, which makes us able to study both local and global information.

2.9.1 The convolution operation

Given an image $I(x, y)$, one might want to apply a filter to capture some relevant information or feature, i.e. edges. A way of capturing an edge, is by designing a specific **kernel** $K(x, y)$ (mask), to be applied locally across the image, and resulting in a high response if this specific feature occurred.

To apply the kernel K of size $m \times n$ on the $M \times N$ image I, we use the convolution operation, which typically is denoted with an asterisk:

$$S(x, y) = (I * K)(x, y) = \sum_{x=1}^{m} \sum_{y=1}^{n} I(x - m, y - n) \cdot K(m, n) \tag{2.16}$$

This is done for all pixels in the image of location (x, y). Thus, to apply it on the entire image, the idea is to apply the kernel K on the image I in a sliding window fashion. As the kernel slides across the image, the convolution operation is applied locally for each pixel. This means that for each pixel in the image, the kernel is applied, resulting in an updated pixel. The output S is sometimes referred to as the **feature map** (Goodfellow *et al.*, 2016).

The convolution operator can easily be extended to work in any dimension. The kernel is always some hyperrectangle (n-dimensional rectangle). It is common practice to use symmetrical kernels, i.e. hypercubes. Although, working with a CT-stack (3D) of different resolution in the transverse and longitudinal direction, it is possible to introduce this resolution difference in the kernel sizes.

2.9.2 Pooling

A typical CNN can be split into three stages. First, the generation of features using convolutions is performed, usually parallelized. Then, each result in sent trough a linear activation function. In the second stage, each of these activations, are sent through a nonlinear activation function, which produces an output. This is often referred to as the detection stage. In the last stage, a **pooling function** is used to further modify the output (Goodfellow *et al.*, 2016).

A pooling function replaces the output at a certain location by a summary statistic based on the neighbourhood outputs. This results in a downsampling depending on the neighbourhood size. Examples of summary statistics might be: max, min, average, standard deviation, L^2-norm.

The most common pooling layers are **max pooling** and **average pooling**. In general, max pooling seems to be the most favourable choice, since it is better for extracting more extreme features like edges, which is of most relevance in the case of object detection (Tompson *et al.*, 2015). Others claim that average pooling is the better choice in general, because it encourages the network to look for discriminative regions of the entire object, compared to max pooling (Tompson *et al.*, 2015).

Pooling helps making the representation invariant to smaller translations in the input. To quote Goodfellow *et al.* (2016): "*Invariance to local translation can*

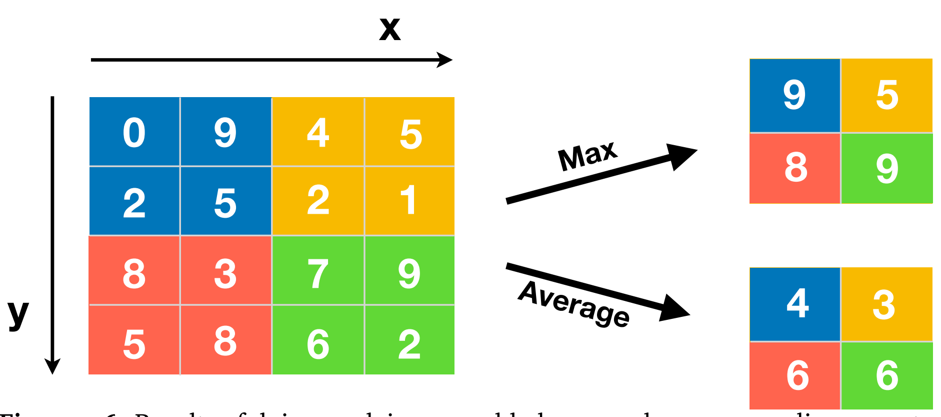


Figure 2.6: Results of doing applying unpadded max and average pooling operators with kernel size 2×2, on the same input (for even sized kernel widths neighbourhood designs may vary).

be a useful property if we care more about whether some feature is present than exactly where it is". This holds true for object detection and classification.

Using pooling layers, it is possible to produce local invariance to *any* transformation. This is done by pooling over separately parametrized convolutions of output. This is very powerful, since it can be easily adapted to any transform and problem.

2.9.3 Complexity problem

A problem with CNNs, is that they are computationally expensive. Thus, it is of interest to apply methods that make CNNs more efficient, without degrading its performance. One concept is called **strided convolutions**, which consists of downsampling the convolution function. In strided convolutions, the sliding window approach is applied, but only at every n'th convolution. This reduces the ability to capture fine edges. The step size n of the sliding window is referred to as **stride**.

Another way to decrease the amount of computations, is to use **tiled convolutions**. Instead of learning weights for every single position in the input, we learn kernels that we rotate as we move around the space. That way we get a compromise between local and global information, and the method is much less computationally expensive.

2.9.4 Dropout

In any neural network overfitting must be handled in order to achieve optimal results. There are many ways of accomplishing this, and one of these is to introduce a regularization method called **dropout** (Srivastava *et al.*, 2014). Using dropout, a neural network is regularized by adding noise to its hidden units (Li *et al.*, 2018), and it is a extremely simple and computationally inexpensive to use.

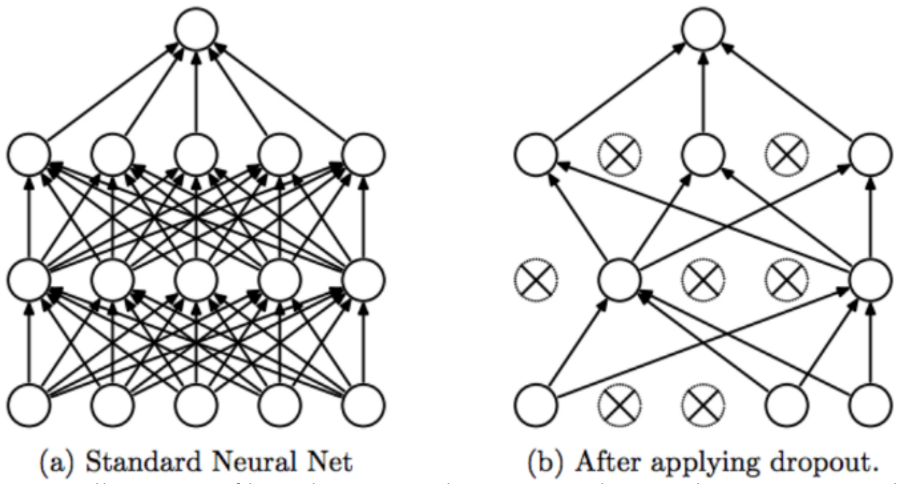


Figure 2.7: Illustration of how dropout works on a neural network (Srivastava *et al.*, 2014).

Each hidden activation is multiplied by a Bernoulli distributed random value of some pre-defined probability p, often referred to as dropout rate. This results in the updated activations:

$$\hat{x}_k = a_k \frac{1}{p} x_k \tag{2.17}$$

where x_k corresponds to activation k, $a_k \sim Bernoulli(p)$, and p corresponds to the probability of dropping a hidden activation. During training, activations are randomly dropped for each batch, resulting in the network not being able to use all of the weights during training. This makes the network less prone to overfitting, since it will be forced to "re-evaluate" past decisions.

Dropout can also be applied in all convolution layers, since it contains learnable weights. This is quite commonly done, and results in better performance than using them only in fully connected layers for CNNs (Srivastava *et al.*, 2014).

Note that dropout should only be applied in the training phase, since it adds noise and degrades performance. Therefore, during inference hidden activations would be simply: $\hat{x}_k = x_k$.

For images, dropout can also be understood as randomly dropping individual "pixels". The problem using this for images is that adjacent pixel are highly correlated, and therefore using dropout might introduce too much noise. The solution is to use **spatial dropout** (Tompson *et al.*, 2015). The idea is to randomly drop entire feature maps, instead of random "pixels". This means that morphological features are less disrupted. Therefore, spatial dropout is commonly used in convolutional layers. However, ordinary dropout still is a viable choice for regularizing ordinary neural networks.

2.9.5 Batch Normalization (BN)

During training it is common to divide the training set into mini-batches, as it makes training faster, and works as a natural regularizer. Even though mini-batch training is fast, it is not necessarily efficient. Therefore, a method called **batch normalization** (Ioffe and Szegedy, 2015) has been proposed. The idea is to normalize each neuron dependent on values $x = x_k$ over a mini-batch $B = \{x^{(1)}, ..., x^{(m)}\}$, of m instances (Li *et al.*, 2018). The normalization is defined as:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \qquad \sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)} - \mu_B\right)^2 \qquad z^{(i)} = \frac{x^{(i)} - \mu_B}{\sqrt{\sigma_B^2 + \tau}} \tag{2.18}$$

Note that in the multi-dimensional joint case, what might happen is that it results in singular covariance matrices. Thus a smoothing coefficient τ is needed (Ioffe and Szegedy, 2015).

Normalizing activations that depend on mini-batches allow efficient training, but is neither necessary nor desirable during inference. What was proposed instead is to use moving averages of neural means and variances (Li *et al.*, 2018). Thus, during inference, we get the moving-standardized transform:

$$\hat{x} = \frac{x - E^{moving}(x)}{\sqrt{Var^{moving}(x) + \tau}} \tag{2.19}$$

Further details about how $E^{moving}(x)$ and $Var^{moving}(x)$ can be implemented is described in Ioffe and Szegedy (2015).

2.10 Classification

It is possible to train a CNN to classify images by supervised learning as explained in section 2.9. By using convolution layers, features can be extracted from the image, and by using pooling layers, feature maps can be downsampled. By applying a sequential convolution layer, more complex features can be extracted - effectively at different magnification levels. By doing these operations sequentially multiple times, a *deeper* network can be made.

At the end of these layers, dense layers (ordinary neural networks) can be applied. This way, *end-to-end*, the network generates features from the input, and by supervised learning, it learns which features are relevant to solve the specific task. For a classification task, the network will learn which set of features corresponds to each class. In other words, the network learns a "*fingerprint*" for each class. Given a new input, it will predict which class it belongs to depending on how similar the generated features are to these fingerprints.

2.10.1 VGG16

One of the most popular architectures for image classification is **VGG16** (Simonyan and Zisserman, 2015). Using smaller kernel sizes (3×3) and increased depth resulted in significant improvement on the ImageNet data set. They also showed that the network generalized well to other data sets.

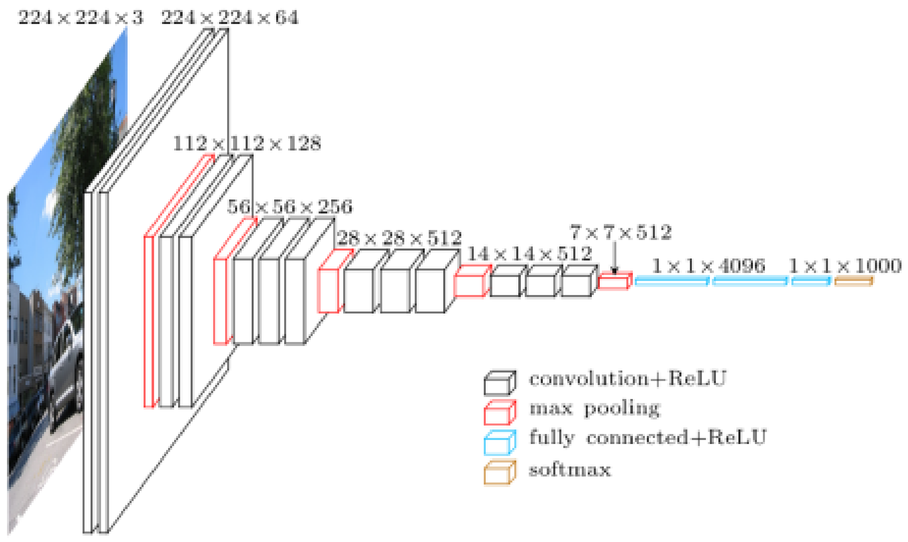


Figure 2.8: Macroarchitecture of VGG16 (Frossard, 2016)

The overall network contains in total 16 layers, as illustrated in Figure 2.5, hence the name. Simonyan and Zisserman (2015) also proposed a 19-layer architecture **VGG19**.

VGG16 takes an RGB-image of fixed size $244 \times 244 \times 3$. It contains a total of five max-pooling layers with kernel size 2×2 kernels (of stride 2), effectively downsampling the feature maps by two, overall five times. Between these, there are multiple convolution layers; two convolution layers at each level before the second pooling-layer and three layers after. The number of convolutions is the same for all convolution layers at one level (magnification level of feature maps) - starting with 64 and doubles for each pooling layer. This is a quite common convolution configuration. The idea is to increase (double) the number of convolutions to compensate for the loss of resolution after downsampling (by two).

In the end, three fully-connected layers are used (3LP) with neuron configuration: 4096, 4096, 1000. The reason for the last layer to have 1000 neurons is because this is the number of classes in the ImageNet data set. In the last layer a softmax activation function is used. After all other fully-connected and convolution layers, ReLu activation functions are used. The original VGG16-architecture *does not* contain any dropout or batch normalization, to regularize the network, but L2-regularization was used. We will not be using L2-regularization in this thesis. However, further information about the regularization method can be found (Krogh and Hertz, 1992).

2.11 Segmentation

When performing segmentation of an image, each pixel is assigned a class. There are simple methods to perform segmentation, but traditional methods require that we are able to define an algorithm to segment the object of interest consistently. Traditional intensity-based methods, i.e. Otsu's method (Appendix B.2.1) and region growing (Hojjat and Kittler, 1998), fail to handle more complex cases. For instance in medical imaging, which often includes noise, artefacts and inhomogeneity.

As discussed in section 2.10, it is possible to perform classification of an image, by extracting features using sets of convolution and pooling layers, and learning which features are relevant using fully-connected layers. This set of layers before the dense layers are often referred to as the **encoder**.

Instead of assigning a class to the entire image, it is possible to use the features map of the bottom layer, and upsample it to have the same size as the input. By doing this, we have assigned a class to pixels, instead of the whole image, and we have effectively performed segmentation of the image. The set of up-sampling layers to produce the final output, is called the **decoder**.

Using only the last feature map(s), the resolution in the segmentation will usually be poor. A solution is to up-sample the image in two steps and using

skip-connections to add feature maps from the bottom layers of the decoder. Skip-connections means extracting data from the encoder and concatenating it with the decoder.

Using a two-step up-sampling in the decoder, using skip-connections from the two bottom levels to each decoder-level, respectively, result in one of the most basic architectures for CNN-based image segmentation called FCN-16 (Long *et al.*, 2014). It was also proposed to use one more up-sampling step in the decoder, with one more skip-connection. This architecture was the best forming one of these, and is called FCN-8 (Long *et al.*, 2014).

These types of down- and up-upsampling CNN-based architectures, that produce an output of the same size as the input, have therefore been given the name **autoencoders**. The job of the encoder is to compress and extract relevant information from the input, and the decoder uses these to produce a conclusion for every pixel in the original input.

By using these types of network, we can perform **semantic segmentation**, where each pixel is denoted a class. It is also possible to perform **instance segmentation**, which is when in addition to each pixel being given a class, an object class prediction is performed. This means that the boundary of a group of pixels is also predicted, effectively segmenting sets of pixels, instead of just independent pixels.

As an example, consider detection of pedestrians in an image. If you use semantic segmentation, you classify which pixels correspond to the pedestrian class, independently of the others. That way if two people are adjacent, using semantic segmentation, you cannot learn that these are two different pedestrians. Using instance segmentation, you not only perform segmentation, but you also perform object-detection. One of the most popular instance segmentation networks is called Mask R-CNN (He *et al.*, 2017). These types of networks are extremely computationally expensive. It is not feasible to perform 3D instance segmentation using these state-of-the-art architectures for current high-end GPUs, even though these are the ideal candidates for many 3D object detection tasks, e.g. lung nodule detection.

2.11.1 UNet

A popular segmentation architecture, especially for medical imaging, is UNet (Ronneberger *et al.*, 2015). It is similar to the FCN-8 architecture, but handles the decoder differently. Instead of simply up-sampling the feature maps, it introduces deconvolution (also called **transpose convolution** layers, which learns how to best up-sample the feature map to produce optimal segmentation. It also uses skip-connections, but extract feature maps across from the encoder at each respective level, in a symmetric fashion. Similar to VGG16, it generalizes well, and it has been quite promising, especially in medical image analysis.

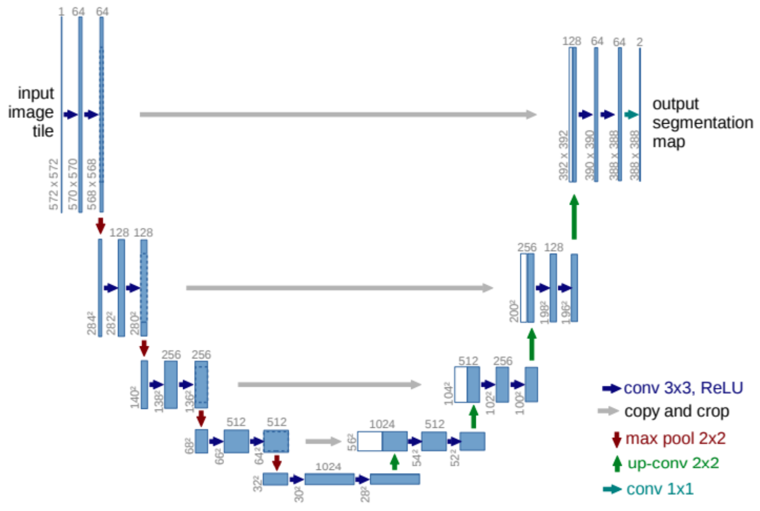


Figure 2.9: UNet architecture (Ronneberger *et al.*, 2015)

The original architecture, input grayscale images of size 572×572. At each level it contains two **unpadded convolution** layers. Similarly to VGG16, it uses kernel size 3×3, ReLu activation functions after convolution layers, and 2×2 max-pooling operations with stride 2. It also used the same convolution configuration, starting with 64, doubles each time, but goes one layer deeper, effectively stopping at 1024. Note that using unpadded convolutions, the image size is downsampled for each *convolution* - due to the border not being included (since it lacks valid pixels outside the image).

The idea with UNet was to also learn how to improve up-sampling of data to achieve optimal segmentation. Up-sampling was done symmetrically as in the encoder, resulting in a symmetric autoencoder, hence the name *U-Net*, or UNet. To enhance resolution in the decoder, information from the encoder was copied across and merged (concatenated) with the decoder feature maps. This operation is referred to as **(residual) skip-connections**, which has the same interpretation as ordinary skip-connections.

The decoder is different from the encoder in the convolution configuration, but otherwise it is symmetric. In the end, it outputs a segmentation map using a softmax activation function.

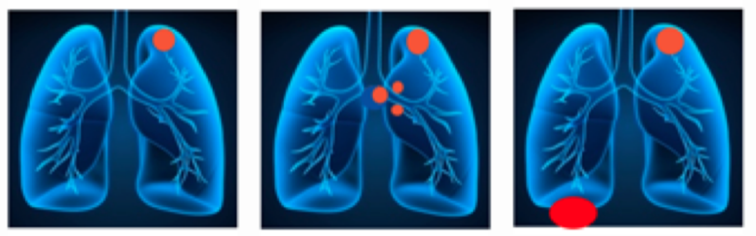
Part I

Lung cancer

3

Clinical motivation

Whereas early stage lung cancers have good prognosis, the chance of survival drops dramatically if the disease has spread to lymph nodes or other organs. An early, precise diagnosis of nodules with cancer is therefore crucial to offer curative treatment and improve lung cancer survival. Several National Health authorities have applied fast-track patient care programs for investigation of lung nodules that may be cancer. The time from nodule detection in CT to clinical decision-making is used as a quality marker for the health care services, which gives another reason why a fast and precise CAD for lung nodules is called for.



Stage I-II	Stage III	Stage IV
50 ± 20%	15 – 20%	< 1%

Figure 3.1: Clinical stages of lung cancer and their respective 5-year survival rates below (The Norwegian Lung Cancer Group (NLCG) Guidelines for Lung Cancer Management, 2017 (nlcg.no). Figure: H. Sorger

The vast majority of early lung cancers occur in the form of a nodule, defined as a round or oval lung lesion < 30 mm in diameter and completely surrounded by lung tissue. Lung nodules occur in up to 33% of CT scans in a screening population of aged smokers (Callister *et al.*, 2015), and in 13% of unselected CT scans. The prevalence of malignancy in patients with lung nodules however, is lower, varying between 1-12% depending on the study population and nodule characteristics such as size and a number of morphological features in CT (Wahidi *et al.*, 2007).

The main clinical challenge is therefore to recognize as early as possible the nodules with the highest risk of malignancy that should be subject to removal or close CT monitoring. Equally, low-risk nodules should be dismissed from follow-up to save health care resources, avoid futile investigations and patient concern (van den Bergh *et al.*, 2008, Leung and Smithuis, 2007).

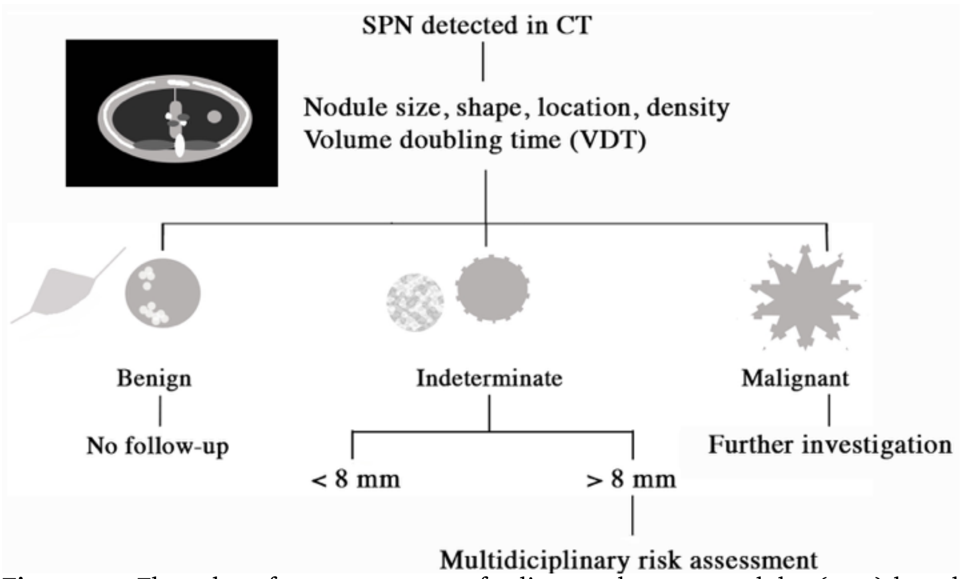

SPN detected in CT
Nodule size, shape, location, density
Volume doubling time (VDT)
Benign
No follow-up
Indeterminate
< 8 mm
> 8 mm
Multidiciplinary risk assessment
Malignant
Further investigation

Figure 3.2: Flow-chart for management of solitary pulmonary nodules (SPN) based on CT characteristics. Adapted from The Fleischner Society Guidelines for management of solitary pulmonary nodules, 2017. The Norwegian Lung Cancer Group (NLCG) Guidelines for Lung Cancer Management, 2017 (nlcg.no). Indeterminate nodules < 8 mm size are normally subject to CT follow-up. Figure: H. Sorger

For the remainder of this chapter, we will give some theoretical background on lung nodules and malignancy, as well as the imaging modality used to generate the images we will be using for this problem.

3.1 Lung nodule and malignancy

The most important characteristics a radiologist studies when classifying nodules as benign/malignant are:

- Nodule size
- Nodule shape
- Nodule location
- Presence and distribution of calcification
- Nodule volume doubling time (requires follow-up data). This normally takes 100-400 days for solid cancers, significantly longer for ground glass opacities.
- Nodule density (solid, part solid, ground glass opacity)

In a study by Leung and Smithuis (2007), involving 3356 lung nodules, one of the aims was to find the probability of a lung nodule being malignant as a function of nodule size. A strong correlation was found between size and malignancy in high-risk patients. Summarized results can be seen in Table 3.1.

Table 3.1: Probability of malignancy in patient with high risk of lung cancer (Leung and Smithuis, 2007).

Size [mm]	Total	Malignancy [%]
< 4	2038	0
$[4, 7)$	1034	1
$[7, 20]$	268	15
> 20	16	75

Nodule size is therefore an important malignant predictor. Also, for all 2038 patient cases, all nodules smaller than 4 mm were benign, implying that such nodules are not of immediate diagnostic interest.

Given uncertain findings, it is quite common to do a *follow up scan(s)*. Note that this is mainly of interest if one wants to study how these candidates develop over time - typically 4-6 months later (Kanashiki *et al.*, 2012). Even though a nodule might be small at the time of detection, a follow-up CT scan might still be of interest to study how its size and shape evolves over time. The radiologist would look for the volume **doubling** time, or nodule growth rate since the last CT scan. Solid, malignant nodules typically have volume doubling times of 100-400 days, although signficantly longer for ground-glass opacities, representing invasive adenocarcinomas.

Another indication of malignancy is nodule heterogeneity. A nodule may for instance have a ground-glass component (GGC), or a ground-glass halo around a solid or mass. Malignant nodules often have a solid part, but also a web-like structure. From the study by Leung and Smithuis (2007), 64% of nodules displaying a solid and ground-glass component proved to be malignant. For FFCs the malignancy prevalence was 18%, for nodules with part-solid components 7%. Therefore, heterogeneity is a quite important factor in classification of malignancy. Any GGCs detected is not necessarily malignant, but the chance of it being so is significantly higher than for ordinary lung nodules.

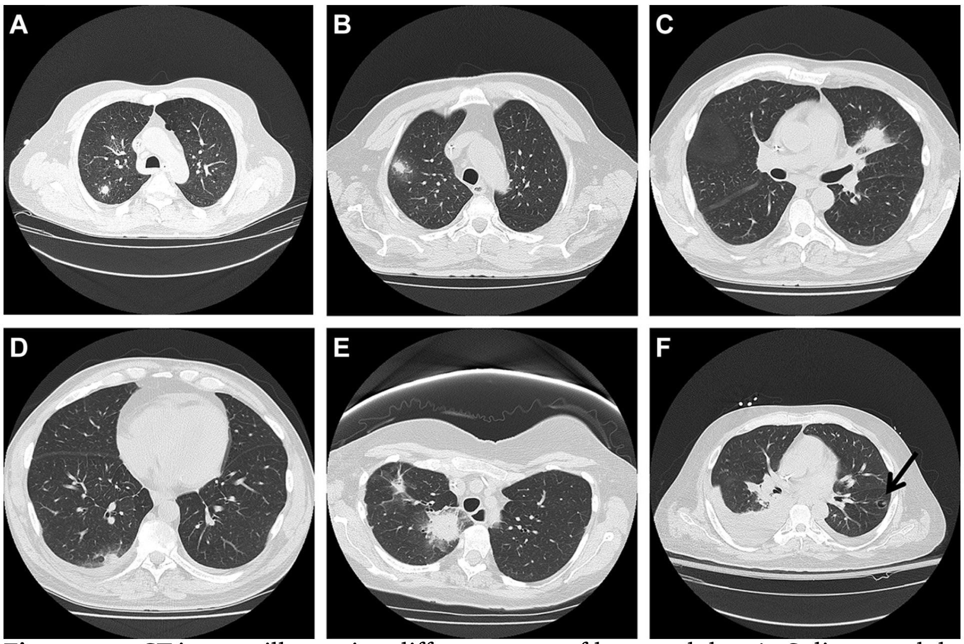


Figure 3.3: CT-images illustrating different types of lung nodules. A: Solitary nodule. B: Solid with ground-glass halo. C: Mass with ground-glass halo. D: Juxta-vascular nodule with pleural effusion. E: Larger solid with part ground glass and irregular speculated density, also smaller but more speculated lesion. F: Cavity nodule (Wingard and Jantz, 2012).

Many biological aspects are beyond the scope of this thesis report. We will only cover the topics we find relevant for detection, segmentation and malignancy prediction of lung nodules. One important anatomical change of relevance is *emphysema*, or destruction of normal pulmonary tissue with resulting air filled gaps and fibrosis. Emphysema is normally caused by cigarette smoking, and this is partly why the condition is associated with a significantly higher risk of lung cancer (Hohberger *et al.*, 2014). In CT, emphysema stands out as homogeneous, dark spots (low attenuation) inside the lung. Even more relevant for this project, the characteristics/morphology of lung nodules may change depending on emphysema severity. This is challenging to include in any CAD-system.

3.2 X-ray imaging

The technology behind the CT-images we observe are based on radiation. The image is created in a process of several steps. First one needs an X-ray tube, which is a vacuum tube with a cathode and an anode. By charging the cathode, high-speed electrons begin to fire towards the anode, then quickly de-accelerate as they hit the nucleus of the anode. This results in energy being released in form of X-ray radiation (Gonzalez and Woods, 2010a).

To penetrate the body, one would need a sufficient voltage for the energy of the X-rays, but at the same time minimizing the dose and maximizing the image quality. As the X-rays penetrate the body, they will interfere with different objects in the body, and it is these scattered rays we want to capture to create an image. One way to capture the rays, is using a phosphor screen that converts X-rays into light. Then the light signal is digitized, and based on the amplitudes, one can predict which elements it has passed through, to create an image.

Note that lung nodules below 5 mm size are rarely detected in conventional X-rays, and the number of false negatives is high.

3.2.1 Computed Tomography

One of the most important imaging modalities in the detection and evaluation of lung nodules down to 1-2 mm size is computed tomography (CT). It uses the concept of X-ray imaging with back projections to create a 2D-image.

The idea is to apply ordinary X-ray imaging from different angles around the object of interest, then capturing the rays in a small detector, and back projecting the resulting photons. By doing so over several angles, one can add/integrate over the resulting back projections, which results in a *sinogram*. This transformation is called the *radon transform*. Then to get the resulting 2D-image, as we observe as a CT image, we apply the inverse radon transform.

All the steps can be summarized as follows:

1. Compute 1-D Fourier transform $\boldsymbol{G_\theta}(\omega, \theta) = \mathscr{F}(g_\theta(x, y))$ of each back projection (or angle θ)

2. Apply a window function $\boldsymbol{w}$ on $\boldsymbol{G_\theta}(\omega, \theta)$, to reduce ringing artifacts. The most common window function is the *Hanning window*

3. Apply the inverse 1-D Fourier transform of the filtered back projection $\Rightarrow f_\theta(x, y) = \mathscr{F}^{-1}\{\boldsymbol{G_\theta} \cdot \boldsymbol{w}\}$

4. Integrate over $f_\theta(x, y)$ for all back projections. This results in the reconstructed image $f(x, y)$

In the continuous case, we could formulate the reconstruction by a single equation:

$$f(x, y) = \int_{\theta=0}^{\pi} f_\theta(x, y)\, d\theta = \int_{\theta=0}^{\pi} g(x cos\theta + y sin\theta, \theta)\, d\theta \tag{3.1}$$

where $g(\rho, \theta)$ is the sinogram and $\rho = x cos\theta + y sin\theta$.

This concept can be used to create images of a volume instead. There are several ways of doing this. A simple approach is to apply the 2D scanner at different increments of a specified range. Then stacking these images, or using some kind of reconstruction method, results in a CT image stack of the 3D volume. The problem with this approach is that there is a trade-off between resolution and dose which is difficult to control. Therefore, what is commonly done is to do a helical scan. Using a spiral scan motion, the 2D scanner is applied continuously in a specified range. Then one can reconstruct the volume as a 2D image stack since one control the spiral motion. This results in reduced dosage, while achieving similar resolution.

3.2.2 Hounsfield Units

X-rays penetrating the body interfere with a medium, causing X-ray attenuation that is used for image creation. There is a significant difference in output amplitude whether the ray has penetrated human tissue, air or bone. By experiments one have been able to find attenuations coefficients μ for a lot of different relevant mediums.

Because the most common medium in the human body is water, this attenuation in water is used as reference. If the measured signal is lower than expected (amplitude for water), it would indicate that the ray has passed through some other tissue. Since the attenuation coefficients of all relevant mediums are known, we could try to map the light signal amplitudes to some values that have some solid physical interpretation.

This is what we call **CT-numbers**, and these are:

$$CT_{number} = \frac{\mu_{Tissue} - \mu_{H_2 0}}{\mu_{H_2 0}} \times 1000 \tag{3.2}$$

By using water as reference, the CT_{number} of water is defined as zero. More dense mediums, such as soft tissue or bone, have positive CT, and vice versa. These are what we call **Hounsfield Units**(HU).

Inside the body, the HU's approximate range is (-1000, 3000), where -1000 is for air, and 3000 for dense bone. Inside the lung we typically only have HU-values in the range (-1000, 1000), but at times one may observe intensities larger or smaller, due to different artefacts in the imaging modality - one of which is salt and pepper noise. However, there are disagreements of the limits in literature. Some say that the limits are in the range [-1024, 1024].

The various intensities in CT images actually have a physical meaning, and any image preprocessing should not alter with these values. Another consequence is that segmentation of larger objects can be easily generalized since each object displays a range of intensities. This is an argument to why most lung segmentation methods are based directly on these intensities. Using other modalities, one can rarely use these simple approaches.

One could also try similar approaches to detect nodules, but for instance blood vessels have similar HU-values, which results in segments where it is challenging to separate these from nodules. Therefore, more advanced and efficient methods were proposed (Li, 2007). Some of these are based on deep learning, which is the approach most state-of-the-art algorithms are based on (Wu and Qian, 2019).

4

Lung nodule segmentation

In this chapter, we will design and train a CNN-architecture for automatic detection and 3D segmentation of lung nodules for thoracic CTs. This will be done in the following steps: data acquisition and pre-processing, design, evaluation and results.

4.1 Data acquisition and pre-processing

4.1.1 The data set

The Lung Image Database Consortium image collection (LIDC-IDRI) is one of the largest lung-nodule-specific databases available, consisting of both diagnostic and lung cancer screening thoracic CT scans with annotated lesions (Armato III *et al.*, 2011). Since it was made public in 2010 with 399 cases, the number of cases has grown up to 1018 - with 1010 patients (Armato III *et al.*, 2015a).

A common clinical challenge is that the smaller the nodule, the harder it is to provide a tissue specimen by endoscopic or transcutaneous methods. The malignancy potential of a tumor candidate must therefore often be determined from radiology images. The same is true for tumor volumes. It is challenging to find the true boundaries of a lung nodule.

Therefore, the ground truth has been based on the opinion of four expert radiologists through a two-phase reading process. First there is the blindfolded phase, where each radiologist independently has to annotate what he/she believes is a nodule. If the nodule is in the range (3,30) mm, he/she has to draw the 3D-contour of the area believed to be a nodule, by drawing the boundary for each image in the stack. If the nodule is smaller than 3 mm, only the location of the approximate 3D center of the mass is marked.

In the second phase, all conclusions are compiled, and each radiologist can compare conclusions with the other's opinions. Next, the radiologist can independently draw a final conclusion. This means that four ground truths exist for each patient case. Ideally, one would have only one, but if the radiologists were forced to come to a single conclusion, actual malignant nodules might have been eliminated. Therefore, to keep as much qualitative information as possible, there are four ground truths.

Not only did each radiologist had to mark segments, but he also had to label nine different attributes of the nodules with a degree of five. The attributes are: *sphericity, margin, calcification, internal structure, spiculation, subtlety, lobulation, radiographic solidity* and *malignancy*.

The data set contains a total of 2660 nodules with great variation in size, shape and malignancy degree. Also, for each scan there is different imaging quality, which results in noise and other artefacts. This means that the networks or algorithms based on this data set, should be quite general.

All scans are capturing the entire lung region, and each image has been scaled to have the dimensions 512×512. Resolution-wise there is a large difference between the patients. In general, image resolution is approximately 0.7×0.7 mm^2, always the same across axes, while in the longitudinal direction it is typically 1.5 mm. Note that especially longitudinal resolution varies a lot in the data set, due to different imaging configurations of different scanners used by the operator. All CTs use Hounsfield units.

The CT scans are clinical thoracic CT (chest). They are from seven participating academic institutions from around the world, in an ordinary clinical setting - no scan was performed specifically for the purpose of the database. This means that there is a heterogeneous range of scanner models and technical parameters chosen to create this data set, which makes it ideal for designing robust and general nodule CADs.

Each CT scan was initially presented at a standard brightness/contrast setting without magnification. But the radiologist was allowed to adjust brightness, contrast and magnification to enable the most complete interpretation of the scan.

As the 3D-reconstruction of the CTs differ among manufacturers, CTs also differ in terms of contrast. All CTs can be grouped as: "soft" (n=67), "standard/nonenchaning" (n=560), "slight enhancing" (n=264) and "overenhancing" (n=127), where *n* corresponds to the number of scans (Armato III *et al.*, 2011).

The intent was to include only a single CT from each patient in the data set to avoid correlation between scans, but it was later found that there were two distinct scans for eight patients included. The data set originates from a screening study, where patients with a lot of nodules were oversampled. This means that the average number of nodules might not reflect the expected number of nodules in a typical screening scenario.

The data available for each patient are DICOM-files, with corresponding XML-file. The actual 2D CT-images (image stack) is stored in the DICOM-files, including also other imaging parameters (metadata) used in the scanning. The XML-file contains all the relevant information about the nodules from the four radiologists' opinions.

4.1.2 Pre-processing

To use the DICOM and XML-files, we still needed to do some preprocessing in order to get the image stacks, ground truth and other relevant metadata information, i.e. resolution. This was done by parsing the DICOM-files to get the volumes, parsing the XML-file to get ground truth, and then saving this information for each patient in the hdf5-format. This format was chosen as it makes it easy to extract some amount of data, without needing to read all of it in memory - it is also very fast.

Since the resolutions differ for each patient, we needed to introduce this information in the data set during training. One approach is to re-scale each volume to have an isotropic resolution, such that a voxel has the dimensions $(1 \times 1 \times 1)\ mm^3$. This results in CT-images of varying sizes depending on resolution between patients. Thus, we propose to only stretch along the longitudinal direction. This was achieved by using 3D spline interpolation, with an order similar to trilinear interpolation (which is quite common to use), with a scaling factor of the longitudinal resolution divided by the transverse resolution (**NB!** *At a late stage it was observed a bug in this implementation. Image resolution was not included in the fraction, only longitudinal resolution. Thus, voxels between patients will not no longer be "perfectly" isotropic, but the effect might be negligible, as transverse resolution does not vary that much in the data set.*)

Using Neural Networks, we have to be cautious using too large values in the input, because it can result in stability issues - exploding gradients. Thus, for each patient, it is common to scale intensities to the range [0,1] - typically referred to as intensity normalization.

Since intensities in CT are measured in hounsfield units, different structures are defined in respective ranges. The lung region may vary in the range [-1024, 1024]. Lung nodules have an even narrower range. Thus, *prior* to intensity normalization, it is common to do HU-clipping, which is essentially setting all values outside a range, to its closest limits. That way, when normalizing the image, intensity resolution is not lost. It also makes the search region smaller for the network. In addition, it reduces the effect of some imaging modalities, which does not really matter for the network, but traditional intensity-based methods are quite sensitive to this.

Training deep CNNs (DCNNs) is quite computationally expensive, especially in 3D. Therefore, we were forced to divide the CT volume into smaller chunks (for training). We wanted to introduce as much 3D information as possible in training. The maximum we could choose was 64 slices (64 mm), which should capture even the largest nodules in our data set. With this configuration, we could not use the full images, due to memory constraints. Thus, we downsampled all images to 256×256. The CT volume was then divided into non-overlapping chunks. Overlapping chunks were not used as it requires a lot of storage capacity, and the number of chunks we get for training should be sufficient - especially doing additional data augmentation.

Lastly, before we saved the chunked data, we had to choose a single ground truth. The loss function we are going to use, does not handle float input. Thus, we did a common 50% consensus on the radiologists annotations to yield the final ground truth. The most natural and most common approach, is to do a 50 % or 75 % consensus. For instance the LUNA-challenge from 2016 (Setio *et al.*, 2017) used 75 %-consensus. We chose to use a 50 % consensus, having in mind that it was better to get more false positives than to lose possible malignant nodules.

Using a 75 % consensus tend to undersegment some nodules, since the inter-variability in annotations some times tended to be extreme. Using a 50 % consensus-design, occasionally, might result in fragmented ground truth. This was not observed during pre-processing or training, but during evaluation it was observed that in one out of 80 patients, a nodule had been split into three - one larger and two smaller segments. In this case, we used morphological closing (Appendix B.4) to fix the segment, such that the network did not have reduced sensitivity, when all three segments belonged to a single nodule.

4.2 Design

4.2.1 Architecture

The network we have chosen is heavily inspired by an architecture called UNet (Ronneberger *et al.*, 2015). The original architecture was designed to handle 2D-images. What we have done is to generalize it to handle volumetric data. Note that this modification has been done before by several others (Milletari *et al.*, 2016, Shi, 2018), but our modification has some differences. The overall methodology is the same for our network. The main differences lie in the kernels of the convolution layers and other design choices which we felt would better suit volumetric data, as well as this problem in particular.

Most noticeably, compared to the ordinary 2D-UNet, is the input size and convolution configuration. Current state-of-the-art 3D networks for lung nodule detection and segmentation tasks, use $64 \times 64 \times 64$-sized kernels. Today, we can use larger input sizes, because of the advance in GPU computing power, but still it comes at a cost. Even on high-end GPUs, we still had to reduce the model capacity, and/or depth, and/or batch size using larger input chunks. Although, we hypothesized that even though we had to make a more shallow network, it would still provide better generalization using larger input chunks.

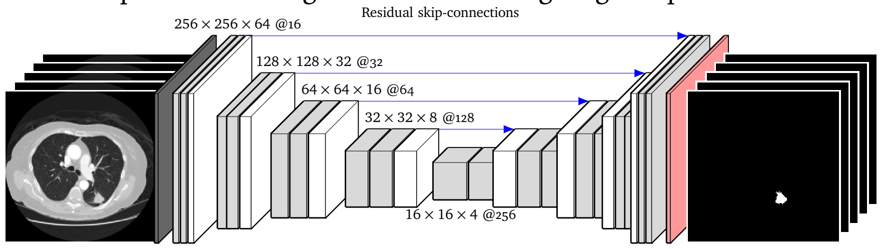


Figure 4.1: Illustration of the architecture used for lung nodule segmentation. It inputs a CT chunk and outputs two-class softmax predictions of lung nodules and background.

First of all we do unpadded convolutions with kernel size $3 \times 3 \times 3$ and $2 \times 2 \times 2$ pooling operators. The network inputs the isotropic (NB: not really, because of error in pre-processing) CT chunks of size $64 \times 256 \times 256$. Similarly to UNet, we use stride 2 and ReLu as activation functions for all layers except the last.

We introduce *batch normalization* after each convolution layer, as proposed in Ioffe and Szegedy (2015), which empirically seemed to boost convergence during training. We also introduce *spatial dropout* with rate 0.2 after each ReLu activation, as proposed in Tompson *et al.* (2015), to reduce the effect of strong correlation between features. Spatial dropout was used in *all* convolution layers.

We used a symmetric convolution configuration, downsampling by two for each max pooling layer, and doubling for each deconvolution/upsampler. Note that we used two 3D convolution layers before each max pooling layer - same kernel size for all.

Since we used one-hot encoding, we used a softmax activation function, which returned probabilities for voxels belonging to each class; nodule/background.

4.2.2 Training

Because of the design choice made, we get M chunks of size $64 \times 256 \times 256$. Within and between chunks, nodules and background were not represented equally, which meant that we had an unbalanced data set problem. Even though one could tackle the problem by weighting the classes within each chunk, there was also a 1:5 ratio of chunks containing just background. It is extremely challenging to handle such 2-level hierarchical unbalanced data sets problems. Therefore, we chose to only train the network using positive samples; chunks containing both nodule and background.

To handle the imbalance within each chunk, we chose to use soft dice loss where DSC was only reported for the nodule class. Then the loss function was only dependent on a single class - effectively balancing classes by neglecting the background class.

During training, Adadelta was used as the optimizer, using default settings as recommended by Zeiler (2012). To solve the problem of overfitting, we monitored validation loss, and the model which produced the lowest validation loss was saved. Methods like early stopping were not used, because convergence was extremely slow. *One single training took 4-5 weeks*. Hence, stopping was done manually. The total number of epochs was 200. After 100 epochs, the learning rate was reduced by an order of ten.

As mentioned earlier, using larger chunks resulted in memory restrictions, and us being forced to reduce the model capacity. Another result was that we were forced to use smaller batch sizes. With the current setup, using a 16 Gb GPU, the largest batch size possible was 2. However, in Masters and Luschi (2018) it was found empirically that smaller batch sizes provide better results in neural networks. Best performance was found using batch size in the range [2, 32].

If not stated otherwise, we used default settings for all functions in the `Keras`-API. Using this architecture, we got a model of ~ 5.9M parameters. `Keras` is a high-level API to build and train deep learning models using `TensorFlow` backend. This also makes it possible to run processes on GPUs.

4.2.3 Data augmentation

Another way of avoiding overfitting, is to have more data. Even though we had 1010 patients, we did not have that many positive chunks after pre-processing. Hence, data augmentation was done. Since we were working with CT-images, augmentation choices were restricted to which kind of transforms would be relevant, or more directly *natural*. For instance, using heavy contrast augmentations would produce skewed intensity distributions, thus perhaps degrading an important aspect in nodule classification.

For each chunk, three transforms were always used, although with random settings. A simple 3D flip-transform along random axis, a random rotation in range $[-20, 20]^\circ$, and translation $[-30, 30]$ in random direction. For rotation and translation, these were only done in the axial plane. This was much faster, and we did not observe any significant boost in performance doing anything more complex. Doing rotation along the longitudinal axis, might result in clipping of huge parts of the volume, especially on the top and bottom of the CT stack. Translation along z was not done, as we did not like the fact of losing whole CT-images by augmentation. It would also produce different clippings of nodules, if located close to the ends along the longitudinal axis. For translation in the axial plane, the lung area was not really near the border. Hence, the same problem did not apply.

4.3 Evaluation

We know that the most common metric for evaluating segmentation is the Dice-coefficient. It is also the most intuitive one. But lets say we have some semi-automatic segmentation algorithm, which needs an initial seed-point on the tumor to start the segmentation. Then it would not make sense to use the ordinary Dice-score metric we have trained our network on. What we would do instead is to have a dice-score metric which only says something about how well the segment is **if** it has found the tumor. Then we throw away the factor of detection, and we can have a metric which fully focuses on segmentation. We adopt the name Dice Score True Positive (DSCTP).

This is the metric we have used to evaluate the segmentation performance. Of course regular DSC is a brilliant overall measure of our detection-segmentation network, but fails to distinguish between detection and segmentation.

For detection, it is common to blindly use sensitivity, specificity and accuracy. The problem is that the input contained a whole CT-chunk, and the objects we are interested in are not single voxels. Hence, we needed to introduce object-specific metrics for detection. This makes sense in our case since detection of nodules, require studying full 3D-objects.

Hence, we made an object-specific recall (sensitivity) metric and false positive rate (FPR) metric to evaluate detection performance. Using a lower prediction threshold, one will accept more voxels as nodule voxels, but at the same time increase FPR. To better visualize this effect, we made a FROC-curve (Appendix A).

From the point clouds, post binary thresholding of the prediction volume, we used connected components with 26-connectivity. Then all nodules in the prediction and in the ground truth, were put into two separate queues. A natural detection criteria is to evaluate hits based on whether the predicted centroid was found inside the ground truth segment. However, this criteria was found too strict, as segmentation annotations are not perfect or consistent - especially not for GGCs.

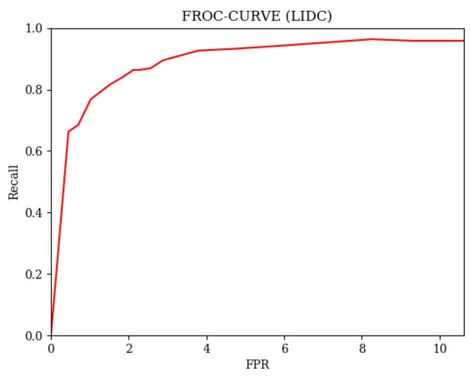
Therefore, we accepted it as a hit if a predicted nodule overlapped with any in the ground truth, or vice versa. Then both overlapping nodules were removed from the queue, to not be double counted, as well as their volume being set to zero in the corresponding prediction and ground truth volumes. Doing this comparison sequentially, also meant that if the predicted nodule overlapped more than one GT nodule, it would only count as one, and vice versa.

4.4 Results

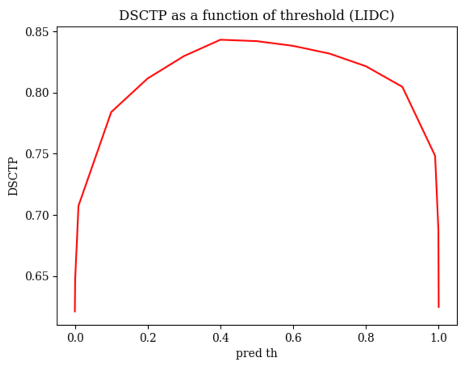
Figure 4.2 (a) shows the FROC-curve, which illustrates the trade-off between recall and FPR as a function of prediction threshold. This is used to evaluate detection performance. Figure 4.2 (b) shows DSCTP as a function of prediction threshold.

Prediction threshold is not "visible" in the FROC curve. However, prediction threshold and FPR are inverse proportional. Thus, increasing prediction threshold decreases FPR, as only accepting voxels with much higher confidence, results in way less voxels being classified as nodules.

However, increasing prediction threshold to lower FPR, results in a decrease in recall. Thus, there is a trade-off between recall and FPR, and that is exactly what the FROC curve illustrates.



(a) FROC



(b) DSCTP

Figure 4.2: Figures illustrating detection and segmentation performance of nodules individually.

By setting prediction threshold $th = 0.2$, we were able to detect > 90 % of the nodules in the corresponding test set, while achieving an average false positive rate of 3.9, which resulted in an average DSCTP of 0.804. Better segmentation performance can be found setting a prediction threshold closer to 0.4-0.5, which also reduces FPR down to 2-2.5, but at the cost of recall.

5 Lung segmentation

In this chapter, we will study two approaches for lung organ segmentation; a traditional intensity-based method and a machine learning approach. Then we will evaluate whether any of these are suitable for post-processing of lung nodules.

In the pipeline, we have not yet used a lung mask to remove redundant information outside the lung, as this is not necessary for training a nodule detector. However, it is possible that the network might generate candidates outside the lung, especially if the FOV is different during imaging than from training. Thus, it is of interest to study whether a lung mask as a post processing step can help reduce FPR.

Traditional intensity based methods for lung segmentation in CT perform quite well. This is because the lung area has quite high contrast compared to the surrounding tissue. Therefore, it is possible to apply simple thresholding techniques. Although, these are easy to use, they tend to struggle in more difficult cases, i.e. if there is a lot of noise (Li, 2007, Wu and Qian, 2019). Machine learning methods have outperformed traditional intensity based methods in many tasks, which also holds true for lung segmentation (Koyuncu, 2018, Mansoor *et al.*, 2015).

5.1 Data acquisition and pre-processing

We did not have access to annotated thoracic CTs. Although, from the *Lunge CT Segmentation Challenge*(LCTSC) in 2017 (Yang *et al.*, 2018), there was annotated cropped (lung-specific) thoracic CTs. This meant that the overall FOV was more narrow than in the LIDC data set. Even though one is able to train a network which performs well on the LCTSC data set, it might not generalize well to the LIDC data set. Since it was not clear if a trained network on the LCTSC data set would generalize to thoracic CTs, we will be comparing a machine learning approach against a traditional intensity-based method.

The LCTSC data set consists of average 4DCT or free-breathing (FB) CT images from 60 patients, but only 30 was made publically available (original training set in the challenge). Three institutions contributed with 20 CTs each. Manual annotations are the same as used for treatment planning in the clinic. All contours were reviewed and corrected if necessary to ensure consistency in the annotations.

CTs were stored in the DICOM-format, and annotations stored in the RTSTRUCT-format. Masks stored in RTSTRUCT, were stored as surface models. Thus, the resulting mask was found after applying nearest neighbour interpolation and applying a binary fill transform.

5.2 Design

5.2.1 Machine learning approach

Traditional methods have proven to be able to segment the lung only based on 2D image information (Li, 2007). This suggests that a 2D-UNet would also perform satisfactory as well. The idea was to use the exact same architecture configurations as for 3D-UNet, but instead of doing 3D-kernels, we will be using 2D-kernels. We also made the network much deeper, which was not possible earlier due to memory contraints in the 3D-approach. While a 3D approach might produce better generalizability, training time was considerably slower for 3D-UNet, and could not be explored due to time constraints.

We chose to apply the exact same pre-processing steps as for lung nodules, except we did HU-clipping on the limits [-1024, 1024], since more global intensity information is relevant for lung segmentation than for lung nodules.

Since training took only a few hours, we did K-fold cross-validation (CV) to train and evaluate the method. Due to the number of patients being quite low,

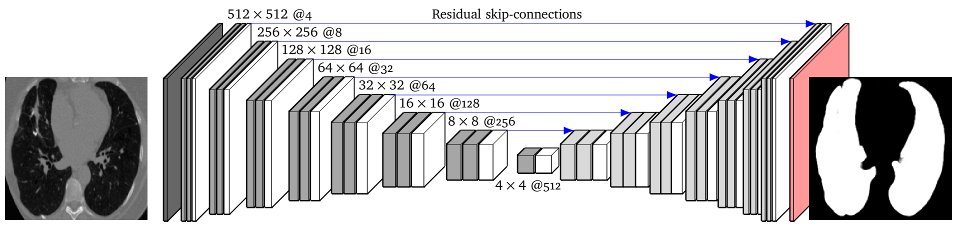


Figure 5.1: Illustration of the architecture chosen. The input is a CT image of specified size in the first layer. The output is a softmax confidence map of two classes; 1 lung, 0 background

30 patients, we chose to do two-split CV. We used five folds, as it produced a natural 80%/20% split between training and test set, and five estimates should give a better indication of the performance on new data.

For augmentation, we used simple random 2D flip and random 2D rotations in the limits [-20, 20]. Since the FOV was more narrow in the LCTSC data set compared to LIDC, we used zoom augmentations with random scaling in the limits [0.75, 1.25], to make it more invariant to different FOVs.

In this case we used Early Stopping during training with a patience of five, since convergence was fast. This implied that if the model had not improved in five epochs, we would stop training. We monitored based on lowest validation loss.

Lastly we used a batch size of 32, which was much higher than for 3D-UNet. It seemed empirically better to use a higher batch size for 2D-UNet on this data set - probably because decision during training would be less random. If it was computationally possible, we would do the same for 3D-UNet. All other training and optimization parameters and choices are the same as for 3D-UNet.

5.2.2 Intensity-based method

To segment the lung it is also possible to use an intensity-based method. We will be combining simple image processing techniques to make a robust method for lung segmentation. A pseudo code of the method can be seen in Algorithm 1. An explanation of all methods used in the algorithm can be found in Appendix B.

The pre-processing was performed in the same way as for 2D-UNet, except we did not interpolate. We used a median filter of kernel size five to improve the

Algorithm 1 Intensity-based method

```
1:  for each image in stack do
2:      HU-clip and normalize
3:      smooth image using median filter
4:      apply Otsu thresholding on central area
5:      filter non-central predictions - circular border
6:      binary hole filling to get lung mask
7:      object removal of specified size
8:      use connected components to generate lung segment candidates
9:      for each candidate do
10:         closing with object removal mid-step
11:         dilate to expand mask - include nodules at boundary
12: connected components on returned mask
13: remove smaller 3D objects using object removal
```

generalization in Otsu thresholding. For the LCTSC data set, we could simply apply Otsu's method directly, but for the LIDC data set, we would first have to crop the central part of the image, removing approximately 5% of each size of the image, and then use Otsu thresholding. This was because the thresholding method was too sensitive to the border of the thoracic CT, and therefore would segment the full body, and not the lung.

Because vessels and nodules inside the lung have similar intensity values as other non-lung-specific objects, one would have to do additional morphological hole filling, to fix the mask. There might be some candidates outside the lungs, especially from the airways. These were filtered away using object removal. The size threshold of 800 was found to be best empirically.

What might happen during hole filling, is that if the lungs are too adjacent, the region between the lungs might also be included. Therefore, to correct for this problem, we used connected components to separate objects in 2D. Then we applied a closing operation with mid-step object removal. Closing was also done to include juxta-vascular nodules, which might have been masked out during thresholding.

Applying this method on all frames, and then doing connected components, produces the resulting lung mask. Still, there are structures outside the lung with similar HU-values, which might generate some additional segments outside the lung. Therefore, an extra 3D object removal was done to filter away any additional 3D segments. Empirically, the size threshold of 5% of the entire volume produced best overall performance.

5.3 Evaluation

To evaluate the methods, we used the LCTSC data set. For the traditional method, lung-specific recall, precision and DSC were reported from all patients. For the machine learning method we did cross-validation, and thus to evaluate the method across all patients, we would only save the predictions from those in the test sets. Overall we got DSCs for all patients from both methods. In order to compare the methods, we wanted to make confidence intervals (CIs) of the point estimates (average DSCs). It was clear that the distribution of DSC seemed skewed, hence, it was of interest to apply a non-parametric method to find these CIs for both methods. On the assumption that patients were independent, or rather drawn iid from some population, we used bootstrapping (Appendix C) to find the BCa intervals for both methods. By bootstrapping patients, we effectively bootstrap the measurements, and hence CIs of the average DSCs (avDSCs) could be found. BCa intervals were also calculated for the other two metrics, including Recall and Precision.

The problem using bootstrapping in the cross-validation(CV) case, is whether or not the bootstrapping assumptions are met. Even though patients are assumed iid, doing CV results in estimates being *dependent of the trained model*. Since the bootstrapping assumption fails, this could produce a bias in the estimation of the variance of avDSC, which might produce an inaccurate CI. However, we did not find another other non-parametric approach to calculate asymmetric CIs, we chose to include the BCa interval(s) regardless.

5.4 Results

5.4.1 LCTSC performance

Table 5.1 shows how well the two approaches perform on the LCTSC data set in terms of segmentation. The point estimates for the machine learning method is all strictly larger than for the intensity-based method. Even though the intervals are strictly larger for the machine learning method, it would be wrong to state that the method performed significantly better, as the bootstrapping assumption failed.

5.4.2 LIDC performance

Figure 5.2 illustrates the difference in lung segmentation performance between both methods. Even though the machine learning method seem to produce finer lung border segments, it fails to include a juxta-vascular nodule, loses some

Table 5.1: Lung segmentation performance of methods evaluated on LCTSC. Point estimates and confidence intervals for all metrics are reported, for both methods

Metric[%] / Method	Recall	Precision	DSC
2D-UNet	99.37 [99.21, 99.45]	99.38 [99.27, 99.46]	99.38 [99.26, 99.43]
Intensity-based	96.50 [95.26, 97.13]	94.42 [92.14, 95.84]	95.42 [93.63, 96.45]

possible candidates, and generates candidates outside the lung. Therefore, the machine learning approach requires the same post-processing techniques used for the traditional method to be useful. The same tendency for both methods was found on all thoracic CTs that were studied on the LIDC data set.

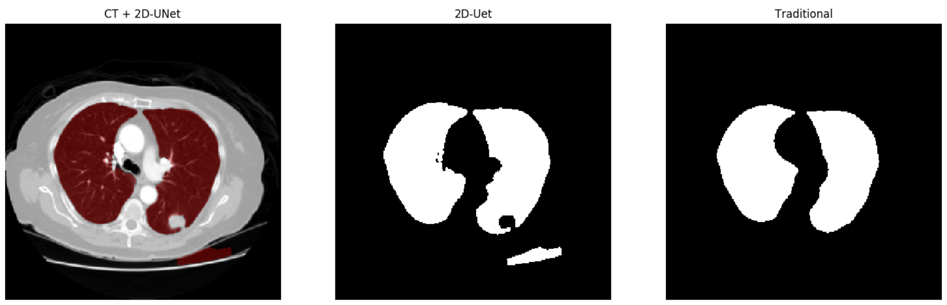


Figure 5.2: Performance between the methods on a thoracic CT from the LIDC data set

Due to uncertainty in generalization and transferability to the LIDC data set of the trained model, we chose to only use the traditional method for post-processing.

6 Malignancy prediction

In this chapter, we will study how we can use CNNs for malignancy predictions of lung nodules, for best integration with the lung nodule detector developed in Chapter 4, using different CNN designs.

The malignancy classifier should be able to work as an independent tool to assist radiologists, making predictions based on initialized centroid or segment. In addition, it could be useful for sorting candidates which need direct clinical invasion, from those to be followed over time. Therefore, the classifier should output the confidence of predictions for different classes, in order for the radiologist to weight classes differently depending on application.

6.1 Data acquisition and pre-processing

As explained in chapter 4, for each nodule in the LIDC data set, there exists a corresponding malignancy prediction. Note that predictions are only (mostly) based on radiographic information. To train a network, we chose to work with a single ground truth. First only nodules from the 50 % consensus design were used. For each accepted nodule, there exist four malignancy predictions as well with predicted scores for each relevant factor. These hand-crafted features are the ones the radiologist used to predict malignancy, but we chose to neglect these, since by using CNNs it should be able to find "the optimal" features itself.

Since the radiologists only used radiographic information (mostly), they might be uncertain in some cases. Hence, instead of giving a binary prediction, malignancy predictions were reported as one of five categories; 1 being benign, 5 being malignant, 3 being uncertain, and 2 and 4 being less confident predictions.

We chose to simplify the ground truth by averaging across the four radiologists' predictions, to yield a single ground truth. This also makes sense since the radiologists tended to be quite consistent, implying that the mean value is a natural estimate for the true malignancy value. An alternative choice is to use the median as it might handle outlier cases better. However, there were only a maximum of four prediction per nodule to work with. Another design choice might might be to only train on cases where the radiologists were highly certain, and discard cases were the radiologists were uncertain. The problem with this choice is that one might end up with a simpler data set, which of course might produce better classification results on the chosen data set, but resulting in worse generalization, since it cannot handle tougher cases.

The main goal of the network is to predict whether the current candidate is malignant or benign, which is a binary classification problem. Hence, two natural design choices were studied. One being to train a network to perform regression and predict which of the categories 1-5 the nodules are assigned to. Note that category 3 is not used for training as the network should not be trained to be uncertain.

The other design choice was to binarize the ground truth to yield two classes directly. If the goal is binary classification, training would be more direct towards this purpose, which also makes the problem easier for the network. Of course, there is the loss of resolution in the ground truth, but at the time of training it was uncertain whether this was relevant. Nonetheless, using a softmax activation function, the output from the binary classifier should mimic a confidence value. Hence, if it predicts malignancy, but less certain, it should output a value in the range (0.5, 1], which would be similar as we would get training with more categories. Therefore, in the end, we chose to only work with the binary classifier, as training with more labels, only seemed to degrade performance.

6.2 Design

The idea was to train a CNN for malignancy prediction. We found nodules larger than 32 mm, even though they should not have been included in the annotations. Hence, we chose to use a 3D kernel of size $64 \times 64 \times 64$, as all nodules should be contained inside this volume. We chose to use a VGG-inspired architecture, but generalized it to handle 3D-data. In the end, we only used two fully-connected layers, with way less neurons, as the model easily overfitted adding more. The last layer contains a softmax activation function, outputting confidence predictions on each class; 0 benign, 1 malignant.

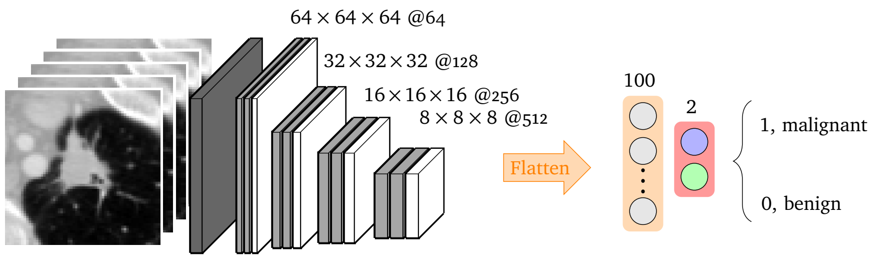


Figure 6.1: Illustration of the architecture chosen for the malignancy classifier - inspired by the VGG16 architecture, but handles 3D input. In the end, there was used two fully-connected layers; 100 hidden neurons in the first one, and two in the output layer, corresponding to the number of classes; 0 benign, 1 malignant.

To perform classification of nodules, we assume that the user or nodule generator has selected a sufficiently centralized centroid, such that the full nodule is included. We will refer to de-centralized centroids as off-centroids. To better handle these cases, we will use data augmentation, to make predictions more invariant to small translations in 3D.

Another design choice is whether we should include nodule segments with corresponding raw CT-data. It is natural that the problem should be easier for the network, since we include the nodule boundary as well. During predictions, we could use our predicted segments as input to the classifier. Given perfect segments, it also handles off-centroid initializations better, since predicted segments will be shifted as well as the raw data.

Ideally, the classifier should not be dependent on predicted segments, because if the network over-/undersegments a nodule, it would probably also affect the classifier. But it was uncertain what the optimal design was. Hence, we studied both.

In `Keras`, it is possible to include segments with raw data during training, by concatenating these volumes - effectively adding segments as an additional channel. This is how training is done with RGB-images for instance, and is a valid approach.

In addition, it might be beneficial for the network to remove "redundant" information around the nodule, which can be done by multiplying the segment with the raw data - effectively extracting only information from inside the nodule. Then one could also only train with the segmented raw data, which should be easier for the network.

Because the number of nodules after the pre-processing is quite low, we chose to use data augmentation to enhance the data set. For all models, we used random 3D 90° rotations and 3D flip. We also experimented with light and heavy 3D translations, random [-5,5] mm and [-10,10] mm, respectively. We used light translations training with raw data only. Including segments, we also tested the method using heavier translations.

Patients were randomly split into three sets; $\sim$ 200 nodules for test, $\sim$ 200 for validation, and the rest for training. The same test set was used for all designs, to make comparison fair, but it was the same split as for nodule segmentation. This was done as the model seemed to generalize better using a larger validation set, and it was also important to have a large test set for evaluation.

We used binary cross entropy as the loss function, since it is ideal for classification, and used the same optimizer as for segmentation. We monitored validation accuracy, which meant that the model which achieved highest validation accuracy was the final product from training. We chose batch size 16, as it seemed empirically to give best performance for classification. In this case, convergence was also slow, hence we chose to stop training manually when it seemed like the network had converged.

6.3 Evaluation

To evaluate the model, we estimated precision(PR), recall(SN) and F1-score (see Appendix A) from the test set only. With this design, there was an imbalance between the classes. Hence, we used a weighted version of all metrics, which is essentially a harmonic mean of the estimated metrics across classes. We also calculated PR, SN and ACC within-class for both (see Appendix A).

To assess uncertainty in the estimates, we used bootstrapping (B=10000) to find the BCa intervals for each respective metric, i.e PR, SN, overall f1-score for both classes. We could **not** bootstrap each individual nodule, as nodules were dependent on patient. Hence, we bootstrapped patients - effectively bootstrapping pairwise labels and predictions for each patient, which satisfies the bootstrapping criteria. A threshold of 0.5 seemed to give best overall ACC. However, in the case of malignancy prediction, it might be of interest for a radiologist to emphasize one class more than the others. It is way worse to say that a malignant nodule is benign, than a benign nodule is malignant, since in the case of sorting, an actual cancer-relevant nodule would have been lost. In the other case, an "unwanted" false positive would have been included.

We tested five different designs for malignancy classification:

1. Classification including segments (ground truth)
2. Raw data only
3. Raw data inside segment only
4. Raw data only + light 3D translation augmentation
5. Raw data + segments + heavy 3D translation augmentation

6.4 Results

Table 6.1 shows the summarized performance results for malignancy prediction for all five designs, evaluated on the same test set. The most important metrics are on the last rows in each sub-table, illustrating the overall precision, recall and F1-score in binary malignancy classification. That is why we only made confidence intervals for these metrics. evaluated on the data set, all intervals overlap. This implies that none of the designs are significantly better than the others in a pairwise comparison. However, the point estimates in F1-score of raw data + segment and raw data only are the highest. For both classifiers, only eight out of 200 nodules were misclassified when evaluated on the same test set.

Table 6.1: Summarized performance results on malignancy classification for all five designs - evaluated on the same test set

Class	Precision	Recall	F1-score
0	0.938	0.991	0.963
1	0.989	0.926	0.957
Overall	0.962 [0.928, 0.981]	0.960 [0.920, 0.980]	0.960 [0.925, 0.980]

(a) Raw data + segment

Class	Precision	Recall	F1-score
0	0.972	0.954	0.963
1	0.947	0.967	0.957
Overall	0.960 [0.926, 0.980]	0.960 [0.918, 0.980]	0.960 [0.926, 0.985]

(b) Raw data only

Class	Precision	Recall	F1-score
0	0.920	0.939	0.929
1	0.941	0.922	0.931
Overall	0.931 [0.890, 0.960]	0.930 [0.881, 0.955]	0.930 [0.890, 0.960]

(c) Raw data inside segment only

Class	Precision	Recall	F1-score
0	0.962	0.885	0.922
1	0.866	0.955	0.908
Overall	0.920 [0.877, 0.950]	0.915 [0.866, 0.945]	0.916 [0.871, 0.950]

(d) Raw data only + small 3D translations

Class	Precision	Recall	F1-score
0	0.789	0.882	0.833
1	0.951	0.906	0.928
Overall	0.905 [0.842, 0.944]	0.899 [0.824, 0.941]	0.901 [0.836, 0.942]

(e) Raw data + segments + heavy 3D translations

7 Lung nodule CAD performance

In this chapter, we will study whether we can boost detection and segmentation performance of the trained network from chapter 4, using different prediction thresholds, prediction designs and post-processing techniques. Evaluation will be done using two data sets, and the goal is to find which design choices and parameters produce the best lung nodule-CAD system in terms of detection and segmentation.

7.1 Data acquisition and pre-processing

7.1.1 Additional test set

To evaluate the complete CAD system, we will be using two data sets. First we will use the independent test set (LIDC), which we have already used in the preliminary studies. Additionally, we will use another independent data set for further evaluation. There are few public data sets available which contain annotations, and the additional we found is from a pre-study of the same people behind LIDC (Armato *et al.*, 2015). The patients should not be the same as in LIDC, but it is natural to think that annotations were done in a similar manner. Similar imaging configurations were also used, which means that it is possible to add the lung mask designed for LIDC, if it is of interest, due to the FOV being similar (thoracic CTs).

The data set origins from a the *LUNGx SPIE-AAPM-NCI Lung Nodule Classification Challenge* (Armato III *et al.*, 2015b). It was part of the 2015 SPIE Medical Imaging Conference (Armato *et al.*, 2015), with support of the American Association of Physics and Medicine, hence we chose to give the data set the name AAPM. It contains a total of 60 annotated patients, where each patient has one or two nodules (73 in total, thus mostly SPN). Each nodule was marked by a centroid, and using both patient and radiographic information, malignancy predictions were made - producing only a single ground truth label. Using this data set, it is not possible to evaluate segmentation performance of the network, but it is much more important to locate nodules, than to segment them perfectly - getting good segments is a bonus.

7.1.2 Pre-processing

Data extraction was done similarly as for the LIDC data set. The only difference was that nodules only contained centroid annotations. For each patient, a binary volume was created. From each centroid, a sphere of radius three was made. It is natural to think that annotation centroids might not be perfect. Thus, we included an uncertainty, which would be less strict on the network in detection.

In practice, a natural choice is to evaluate detection using only centroid annotations. Then we would count it as a hit, if the centroids fall within the predicted segment. However, finding the perfect centroid in annotations is challenging. To counter for this uncertainty we initialized a small sphere of radius 3 from the ground truth centroids. Then we counted a prediction as a hit, if these volumes overlapped, similarly as done for the LIDC data set. However, for this specific data set, there was not found a boost in performance doing this design, but for future work it might be relevant.

7.2 Design

7.2.1 Prediction thresholding

The output from the network is a confidence map of the probability of a voxel belonging to the nodule class. Some minor results were given in Section 4.4, but with the introduction of new post-processing designs, it may produce different optimal prediction thresholds.

Sensitivity is the most important metric for clinicians, as one can accept a few false positives, as long as none important candidates are lost. Changing threshold and some post-processing methods might also affect segments. Therefore, it is of interest to study which prediction threshold is optimum for which set of post-processing techniques.

7.2.2 Lung mask filtering

From the study in Chapter 5, we chose to only use the traditional method as a post-processing step to reduce candidates. Since the mask might not be perfect, especially for juxta-vascular nodules, we chose to do additional dilation of the predicted lung mask. If the nodule was lost in the lung segmentation, it should then be included after this additional step.

We accepted a candidate as a lung nodule, if there was any overlap with the dilated lung mask. We did not want to be too strict, as it was imperative that no actual lung nodules were lost in this post-processing step. We would rather accept more false positives, than to lose any.

7.2.3 Prediction design

Dividing the CT stack in chunks, results in a problem in prediction. Which stride should we use in sliding window predictions? For 3D data, overlapping predictions are commonly used. This is because the network is typically less confident at the boundaries of the chunk, since some structures may be clipped, resulting in some structures also looking like nodules. Doing non-overlapping predictions, it is also possible that a nodule is cut in half. The network would have to predict whether it is a nodule or not, only based on partial information. This might result in loss of nodules or silly generation of false positive.

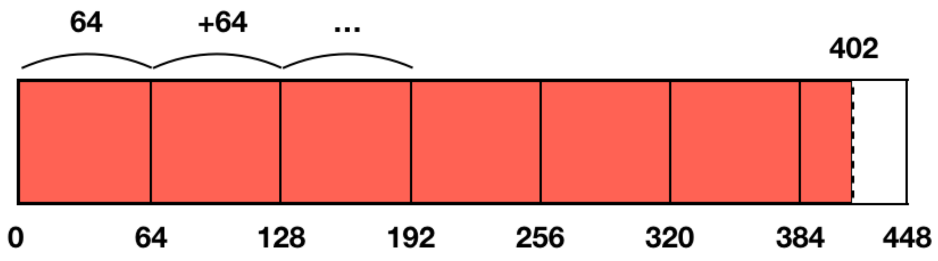


Figure 7.1: Non-overlapping predictions

The easiest and fastest way to do predictions, is to do non-overlapping predictions, which is essentially doing sliding window predictions with stride equal to the chunk size (here: chunk size = 64). It does not require any consensus designs, as there is only one prediction per voxel.

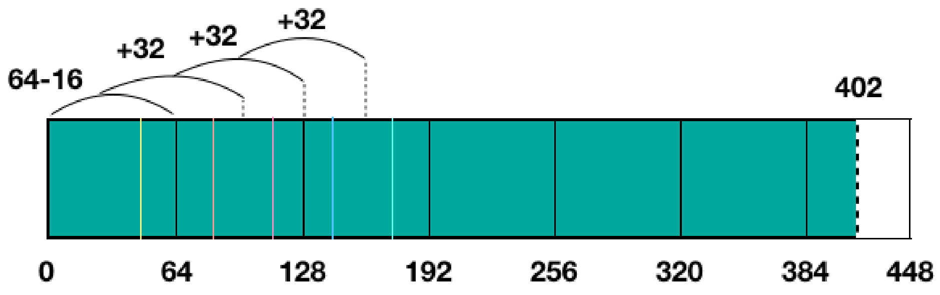


Figure 7.2: Overlapping predictions

A more popular design, especially with smaller chunks, is to do overlapping predictions. The idea is to apply sliding window predictions with stride smaller than chunk size. Using the fact that the network is less confident on the borders, we propose to do overlapping predictions with stride equal to 32, only keeping the middle 32 slices for each chunk. That way, there is only one prediction per voxel in this case as well, and this is only slightly more computationally expensive compared to the non-overlapping approach.

The "most confident result" from this design is to use stride equal to one, but it is highly computationally expensive and slow, and was thus not explored. Using overlapping prediction designs with multiple predictions on the same voxel, also requires some study in which consensus methods is most relevant, dependant on stride. Therefore, we did not focus on this design choice.

Note that for all the prediction designs, if necessary, the remaining slices in the last chunk are zeropadded, prior to prediction, to produce an input of the expected size to the trained network. We will be evaluating both of these designs in our final CAD system.

7.2.4 Predicted size filtering

Given that segmentation performance is satisfactory, the most natural way of reducing FPR is to remove smaller generated candidates. All nodules of a diameter smaller than 3 mm is classified as a small nodule, and is thus not in the training data. Therefore, all nodules smaller than this threshold should be removed. Unfortunately, automatic segmentation of nodules is challenging. If the network has undersegmented a nodule, it is possible that it will be lost with this design. Nonetheless, it only seems natural to remove fragments, if there are any, as they will be counted as an extra candidate.

Since the trade-off is not clear, we will study whether there is anything to gain by filtering candidates dependent on size. We will use connected components with connectivity 26 to separate 3D nodule segments. This implies that we assign a voxel to a an existing class, if any of the closest neighbours in 3D ($3^3 - 1 = 26$) has already been classified. If not, then the voxel is assigned to a new class.

From the labelled volume, the volume is calculated based on the number of voxels in each and every candidate. On the assumption that nodules have a somewhat spherical shape, we will calculate the respective diameter of a sphere for the respective estimated volume, and remove all candidates which have a diameter smaller than some user-defined value.

This makes sense, since radiologists estimate the "diameter" of a nodule by averaging across the largest and smallest diameters in the median plane along all three axes. This is done because they are not able to find the "spherical diameter" as we are from the predicted segments. Thus, our estimate of nodule diameter should be similar to the radiologists'.

7.3 Evaluation

For evaluation of the CAD designs, we made FROC curves, as well as plots illustrating how sensitivity, precision, DSC and DSCTP changed as a function of prediction threshold, and sets of post-processing designs. FROC, sensitivity and precision were used to evaluate detection, while DSC and DSCTP was used to evaluate segmentation (and combined with detection for DSC).

7.3.1 Experiments

We did a range of experiments to find which set of design choices produced the best CAD for detection and segmentation of lung nodules. The relevant designs where: lung mask as a post-processing step, overlapping/non-overlapping predictions and prediction threshold.

For the lung mask and overlapping predictions design, we also studied the effect of filtering candidates based on the predicted size of the nodules.

Additionally, we did a study to unravel how well our model performed on nodules of different sizes. Hence, this is not a post-processing step, but rather a study to see which nodules it might struggle with. This was only done for a single design; lung mask and overlapping predictions.

To further evaluate our method in terms of detection and segmentation, we searched in literature and selected the best performing models which had been evaluated on either LIDC or LUNA, reporting the same performance metrics as us. Based on the observed design choices above, we specified seven models to represent our method, see Table 7.1.

Table 7.1: Seven models with different design configurations

Model	th	th_{pred}	lung mask	overlap
1	0.2	2	yes	yes
2	0.4	2.5	yes	yes
3	0.7	2	yes	yes
4	0.823	2.5	yes	yes
5	0.017	2	yes	yes
6	0.1	NA	yes	no
7	0.2	NA	yes	no

Finally, we show the produced lung nodule CAD system prototype in action, as well as how it is being integrated into the SINTEF developed software CustusX Askeland *et al.* (2015).

7.4 Results

7.4.1 Detection performance

Without filtering on size

Figure 7.3-7.7 show the results of several different experiments. Figure 7.3 shows performance in detection of lung nodules, evaluated on both the LIDC test set and the AAPM data set. In addition to results from chapter 4, we also study recall and FPR as a function of prediction threshold. For all plots, we study how sensitivity and recall change for different designs. It is clear that the overall performance on the AAPM is worse compared to the LIDC data set, as seen from Figure 7.3 (a) and (b).

On the LIDC data set, there does not seem to be much to gain from doing any additional post-processing. We only observe a slight performance increase using lung mask filtering, as seen by the small reduction in FPR (see Figure 7.3 (e)). Overlapping predictions produce slightly worse overall performance, as seen in Figure 7.3 (a). This is because it loses nodules and generates extra

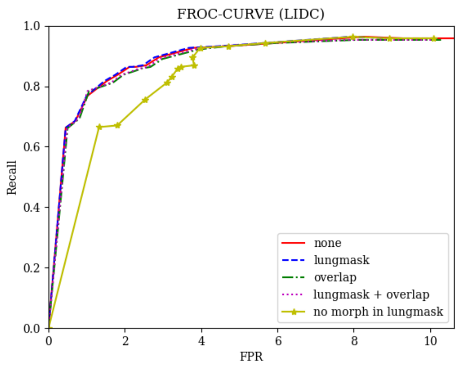
FPs for low-intermediate prediction thresholds (see Figure 7.3 (c) and (e), respectively).

However, introducing a lung mask without proper morphology post-processing greatly degrade overall performance, as seen in Figure 7.3 (a) and (e). This is because it generates many extra false positives.

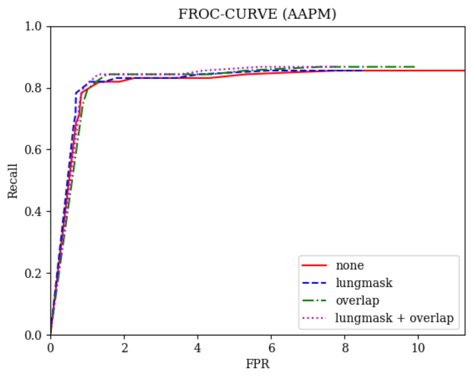
For the AAPM data set, it is clear from Figure 7.3 (b) that using overlapping predictions boosts recall, as all point estimates for all prediction thresholds are strictly larger using overlapping predictions. Also this only give a slight increase in the number of false positives as seen in Figure 7.3 (f). However, using lung mask filtering, greatly reduces the number of false positives.

Looking at both FROC curves, there does not seem to be a significant boost in performance doing any of these additional post-processing steps. However, if any, lung mask filtering shows slight performance enhancement for both data sets. With additional overlapping predictions, there is a slight increase in overall performance on the AAPM data set, but the opposite occurs on the LIDC data set.

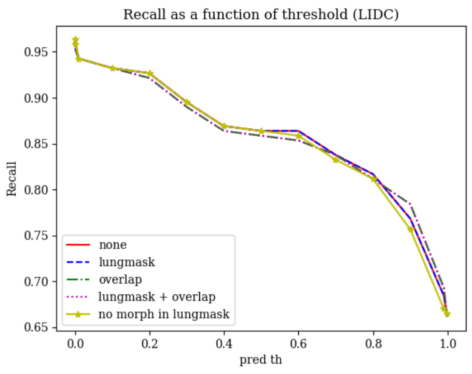
Therefore, overall for both data sets, the best detection performance was found doing additional lung mask filtering with morphology post-processing.



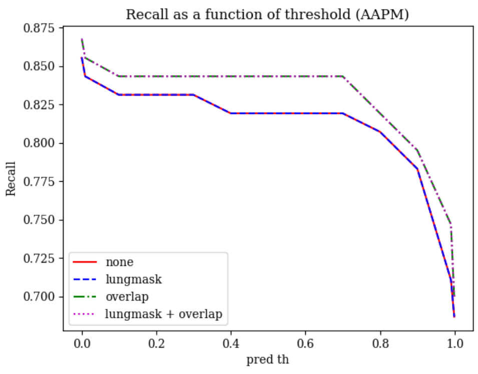
(a) FROC (LIDC)



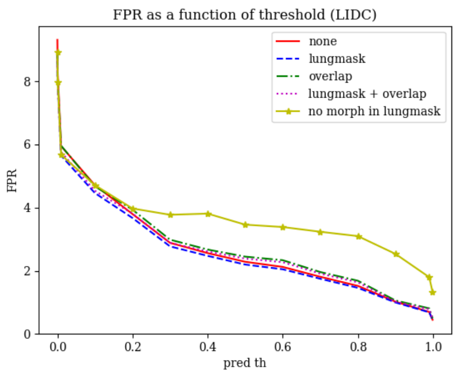
(b) FROC (AAPM)



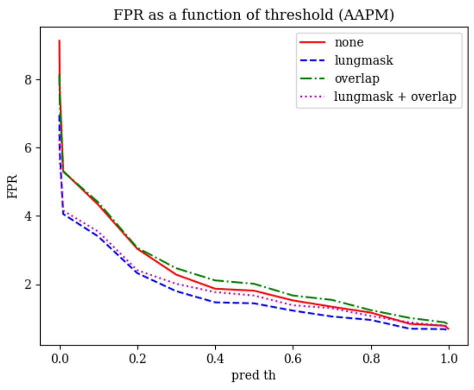
(c) Recall (LIDC)



(d) Recall (AAPM)



(e) FPR (LIDC)



(f) FPR (AAPM)

Figure 7.3: Performance curves for relevant metrics, for different post-processing methods, evaluated on both the LIDC test set and the AAPM data set.

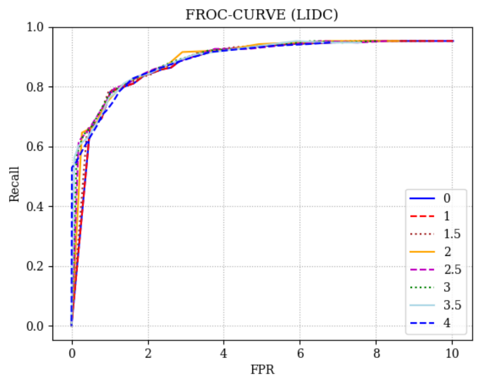
With additional filtering on size

Figure 7.4 shows the results of using overlapping predictions and lung mask filtering with an additional filtering on predicted size. The same set of subplots are reported, as in Figure 7.3, with the same interpretation. However, now the legend show which size threshold was used. Note that these values does not correspond directly to mm, because of the implementation error discussed in section 4.1.2.

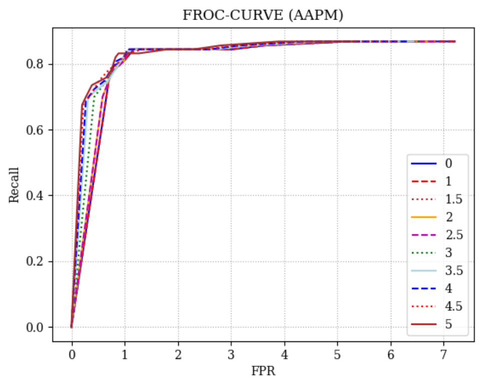
From Figure 7.4 (e) it is evident that filtering predicted candidates depending on size, greatly reduces FPR. This is true for both data sets (see also Figure 7.4 (f)). However, doing heavy filtering on size, also degrades recall, as seen in Figure 7.4 (c) and (d). For the LIDC data set, there seem to be an optimum filtering predicted candidates on size threshold $th_{size} = 2$. This is achieved setting $th = 0.2$, which reduces FPR to 3, with only slight degradation in recall, which produces the jump as seen in Figure 7.4 (a).

For the AAPM data set, we got recall of 0.840 and FPR of $1.85 - 2$, setting $th_{size} = 5$ and $th = 0.2$. However, the best model is found setting $th_{size} = 0.4$ and $th = 0.6$, which yields the same recall, but reducing FPR to 1.1.

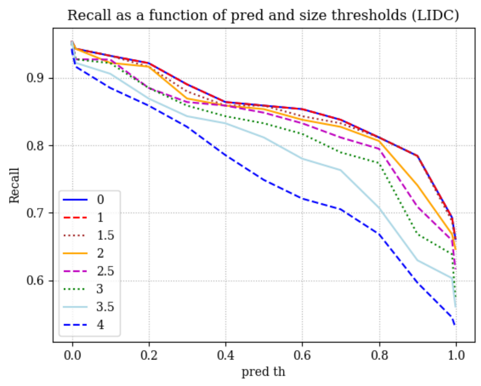
Note that for all FROC-curves, we get a theoretical value of recall equal to 0 at $th = 0$, which explains why we get these weird lines for lower prediction thresholds, as seen in FIgure 7.4 (a) and (d).



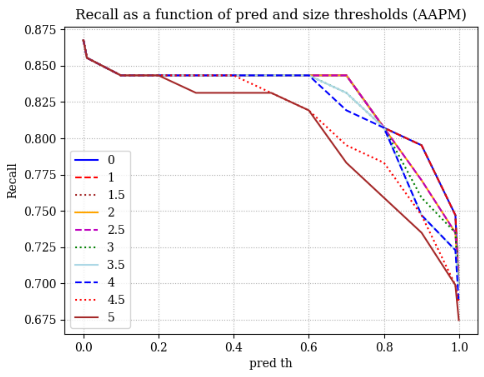
(a) FROC (LIDC)



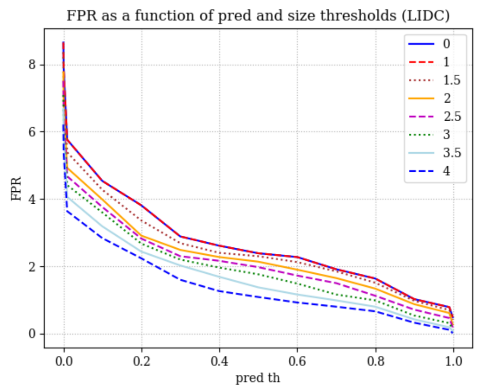
(b) FROC (AAPM)



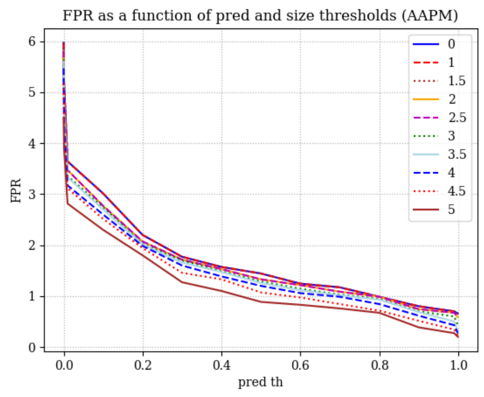
(c) Recall (LIDC)



(d) Recall (AAPM)



(e) FPR (LIDC)



(f) FPR (AAPM)

Figure 7.4: Performance curves for relevant metrics, for one set of post-processing, with additional filtering on predicted size of nodules, evaluated on both the LIDC test set and the AAPM data set.

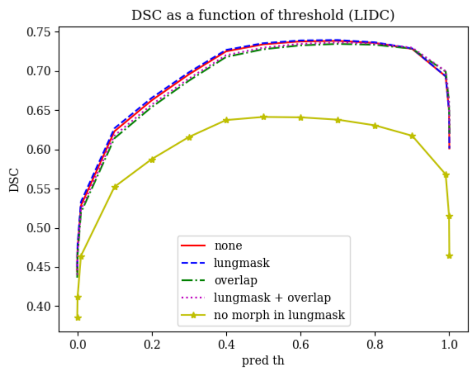
7.4.2 Segmentation performance

Without filtering on size

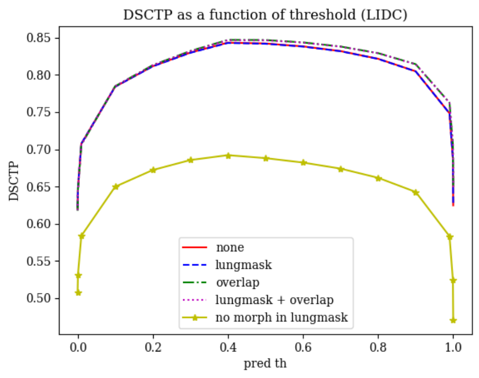
Figure 7.5 shows how DSC and DSCTP varies as a function of prediction threshold, for different sets of design choices. DSC is an overall performance measure which does not directly separate detection from segmentation. Hence, DSCTP is introduced. Note that segmentation is only evaluated on the LIDC data set, as only centroids were annotated in the ground truth of the AAPM data set.

From Figure 7.5 (b), it is clear that the additional lung mask filtering, without the use of additional morphology post-processing, greatly degrades segmentation performance. This is because it includes enough of the juxta-vascular nodules to be counted as a hit, but the nodule is fragmented as a result of an inappropriate lung mask. This shows why the lung mask generated addition FPs, instead of degrading sensitivity, as seen in Figure 7.3 (c) and (e).

For higher prediction thresholds, there is a slight increase in DSCTP using overlapping predictions. DSC is overall lower using overlapping predictions, but this is because it generates more candidates, not because the segments are degrades, as DSCTP shows. As lower prediction thresholds are more relevant for radiologists to maximize recall, there is nothing to gain in segmentation performance adding overlapping predictions, at least in the LIDC test set.



(a) DSC

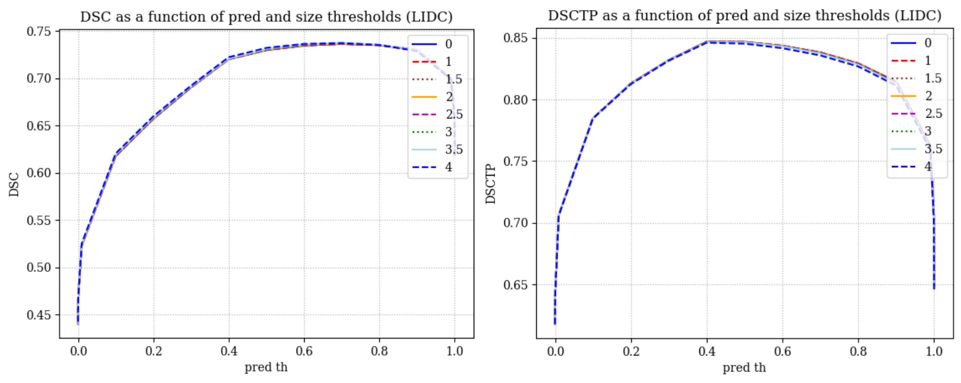


(b) DSCTP

Figure 7.5: Illustrating how DSC and DSCTP varies as a function of threshold (LIDC)

With additional filtering on size

Figure 7.6 shows how DSC and DSCTP change as a function of predicted size threshold. The testing is done the same set of design choices as in Figure 7.4, evaluated on the same test set. As all DSCTP curves remain somewhat overlapping for all prediction thresholds, the optimal th_{size} found in detection, also applies for segmentation. This is an indication that the CAD design seems to segment smaller nodules just as well as larger ones, at least according to DSCTP.



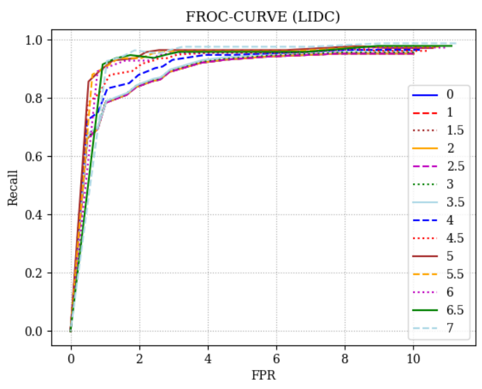
(a) DSC **(b)** DSCTP

Figure 7.6: Illustrating how DSC and DSCTP change as a function of prediction size threshold, using lung mask and overlapping predictions (LIDC)

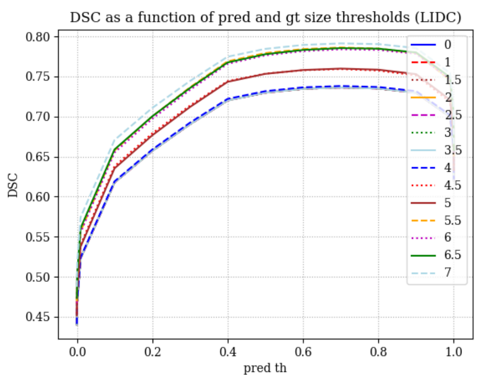
7.4.3 Filtering ground truth on size

Figure 7.7 shows how the same model used in Figure 7.5, performs for nodules of different sizes. The results are achieved similarly as for predicted size filtering, but now filtering is done on ground truth size threshold th_{gtsize}. FPR increases as a result of nodules being removed from ground truth, but not accounted for during prediction, and is thus not as interesting. From Figure 7.7 (b), it is clear that the network performs much better detecting larger nodules. For low-intermediate prediction threshold, recall is as high as 0.975. Since prediction threshold is somewhat more constant for larger th_{gtsize}, it means that the network is more certain in these predictions.

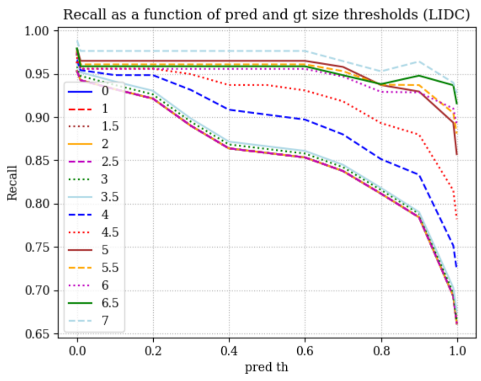
As already seen in Figure 7.6 (b), DSCTP was somewhat independent of th_{gtsize}, but there is a slight increase in performance on larger nodules. From Figure 7.7 (a), it is also clear that the model performs overall better on larger nodules.



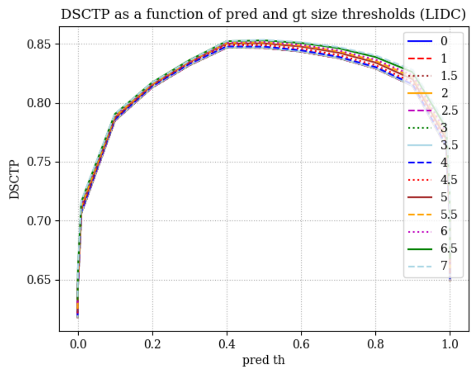
(a) FROC



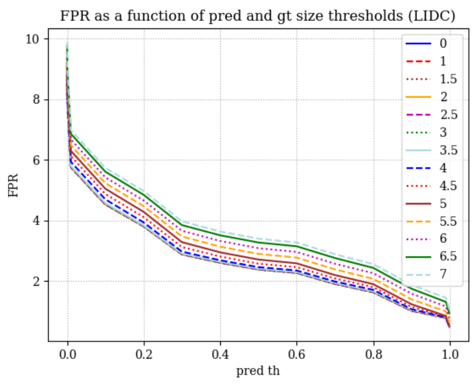
(b) DSC



(c) Recall



(d) DSCTP



(e) FPR

Figure 7.7: Performance curves for relevant metrics, using lung mask and overlapping, studying the effect of filtering ground truth candidates on size (LIDC).

7.4.4 Performance comparison

From Table 7.2, it is evident that our best performing models outperform previous enlisted methods both in terms of detection and segmentation of lung nodules in CT. Most segmentation methods were also semi-automatic, in the sense that they require manual initialization. In these cases DSC is the same as DSCTP. The only other paper using CNNs for lung nodule segmentation is Zhou *et al.* (2018). However, it seems like DSC is only trained and evaluated on $64 \times 64 \times 64$-chunks generated from ground truth centroids. They say nothing about patient reported DSC. Hence, DSC is probably the same as DSCTP in this case as well.

Table 7.2: Performance comparison of our chosen models, against some other modern high-performing designs

Methods	Recall	FPR	DSCTP	DSC
Model1	0.915	2.9	0.813	0.659
Model2	0.859	2.1	0.846	0.722
Model3	0.823	1.6	0.837	0.700
Model4	0.775	1.0	0.824	0.733
Model5	0.941	4.8	0.711	0.529
Model6	0.932	4.5	0.785	0.626
Model7	0.927	3.7	0.812	0.665
Golan (2018) (M1)	0.896	4	NA	NA
Golan (2018) (M2)	0.928	10	NA	NA
Shi (2018)	0.948	14.9	NA	NA
Sakamoto *et al.* (2018)	0.944	4	NA	NA
Jiang *et al.* (2018)	0.801	4.7	NA	NA
Zhu *et al.* (2018)	0.90	5	NA	NA
Zhou *et al.* (2018)	NA	NA	NA	0.772
Mukherjee *et al.* (2017)	NA	NA	0.69	0.69
Wang *et al.* (2017a)	NA	NA	0.7767	0.7767
Wang *et al.* (2017b)	NA	NA	0.8215	0.8215
Messay *et al.* (2015)	NA	NA	0.776	0.776

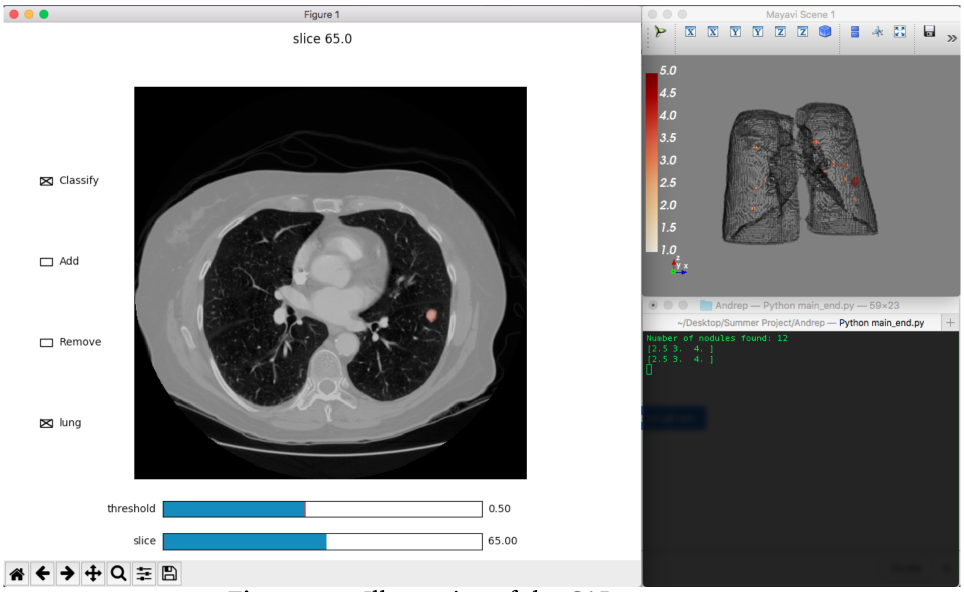
We are also able to achieve object sensitivity of 0.941, while only getting an average FPR of 4.8. Perhaps the best model is Model1 with 0.915 in recall and a FPR of 2.9. DSCTP is also similar as for Wang *et al.* (2017b), which was the method reporting the highest DSC. However, we are able to outperform also this method, at the cost of degrading sensitivity. Model2 produces the highest DSCTP of 0.846, with a sensitivity of 0.859 and FPR of 2.1.

7.4.5 Integration and visualization

As part of the capstone project (preliminary study before thesis), it has been developed a prototype for visualizing lung nodules, as seen in Figure 7.8. This project has not been in focus in this thesis. However, it is part of the on-going development of the lung nodule-CAD system. This prototype was made to illustrate how our model worked, and how it could be possible for radiologists to use the predicted segments in the future.

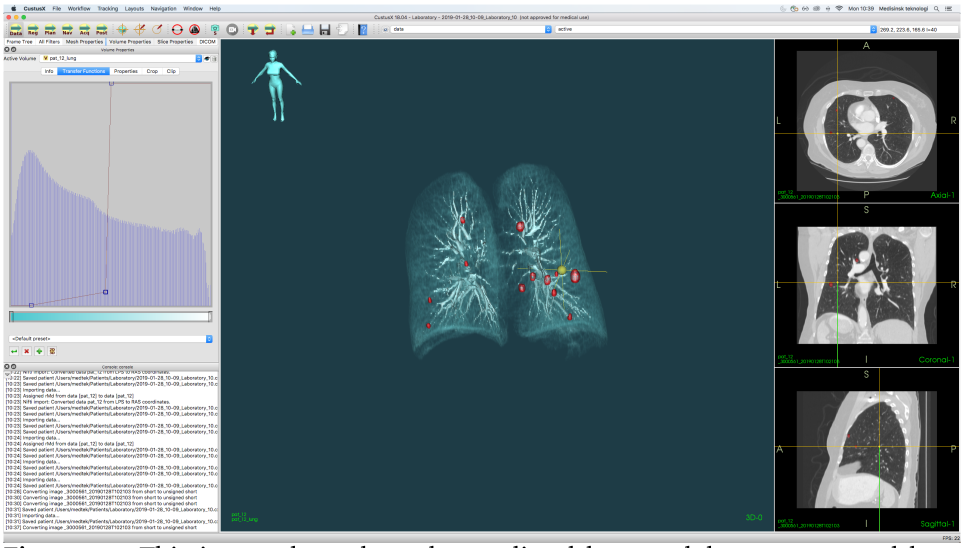
It includes three main components: a 2D slice-wise viewer with generated nodules, a 3D viewer for visualizing the predicted nodules with the predicted lung mask, and finally a command line displaying relevant information when commands are pressed. On the left, there are several options. The two most important options are "lung" and "classify". Pressing "lung" generates the lung mask, to visualize with the lung nodules. Pressing "classify" produces malignancy predictions for all generated candidates. Results of this can be seen in the 3D-viewer, were the confidence range has been scaled to match that of LIDC. We also added the option to remove candidates as seen on the left, as well as add new ones. This uses a semi-automatic segmentation method called level set, implemented based on this paper Márquez-Neila *et al.* (2014). It is also easy to change CT slices using mouse scroll or the slice bar directly, and prediction threshold can be changed using the threshold bar.

Further, we took the concepts shown from the prototype and showed that the same concepts can be easily integrated into the SINTEF developed software CustusX (Askeland *et al.*, 2015). The result, of the same patient as in Figure 7.8, can be seen in Figure 7.9. Notice that the lung segment looks way smoother in this case. That is because the prototype was not built for rendering high-resolution objects. Note that the nodules can be seen in the 2D-viewer exactly as was done in the prototype.



```
Number of nodules found: 12
[2.5 3.  4. ]
[2.5 3.  4. ]
```

Figure 7.8: Illustration of the CAD prototype



```
[10:22] Nifti import: Converted data pat_12 from LPS to RAS coordinates.
[10:22] Saved patient /Users/medtek/Patients/Laboratory/2019-01-28_10-09_Laboratory_10.c
[10:23] Saved patient /Users/medtek/Patients/Laboratory/2019-01-28_10-09_Laboratory_10.c
[10:23] Importing data...
[10:23] Assigned rMd from data [pat_12] to data [pat_12]
[10:23] Nifti import: Converted data pat_12 from LPS to RAS coordinates.
[10:23] Saved patient /Users/medtek/Patients/Laboratory/2019-01-28_10-09_Laboratory_10.c
[10:23] Saved patient /Users/medtek/Patients/Laboratory/2019-01-28_10-09_Laboratory_10.c
[10:23] Importing data...
[10:23] Saved patient /Users/medtek/Patients/Laboratory/2019-01-28_10-09_Laboratory_10.c
[10:23] Saved patient /Users/medtek/Patients/Laboratory/2019-01-28_10-09_Laboratory_10.c
[10:24] Importing data...
[10:24] Assigned rMd from data [pat_12] to data [pat_12]
[10:24] Saved patient /Users/medtek/Patients/Laboratory/2019-01-28_10-09_Laboratory_10.c
[10:24] Saved patient /Users/medtek/Patients/Laboratory/2019-01-28_10-09_Laboratory_10.c
[10:24] Importing data...
[10:24] Saved patient /Users/medtek/Patients/Laboratory/2019-01-28_10-09_Laboratory_10.c
[10:24] Saved patient /Users/medtek/Patients/Laboratory/2019-01-28_10-09_Laboratory_10.c
[10:24] Importing data...
[10:24] Saved patient /Users/medtek/Patients/Laboratory/2019-01-28_10-09_Laboratory_10.c
[10:28] Converting image _3000561_20190128T102103 from short to unsigned short
[10:30] Converting image _3000561_20190128T102103 from short to unsigned short
[10:30] Converting image _3000561_20190128T102103 from short to unsigned short
[10:31] Saved patient /Users/medtek/Patients/Laboratory/2019-01-28_10-09_Laboratory_10.c
[10:31] Importing data...
[10:31] Saved patient /Users/medtek/Patients/Laboratory/2019-01-28_10-09_Laboratory_10.c
[10:37] Converting image _3000561_20190128T102103 from short to unsigned short
```

Figure 7.9: This image shows how the predicted lung nodule segments and lung segments are integrated with the SINTEF developed software CustusX (Askeland *et al.*, 2015)

Part II

Breast cancer

8 Clinical motivation

Worldwide there has been a significant increase in breast cancer incidence over the past decades. In Norway, breast cancer incidence rates have near doubled since the Cancer Registry of Norway started their systematic registration of cancer occurrence in 1952.

Most pathology laboratories are challenged by an increasing number of biopsies each year. At the pathology department at St. Olavs Hospital, Sør-Trøndelag, Norway, the number of biopsies has increased from 35.464 in 2018 to 50.625 in 2018. With this current rate of change, the workload is not sustainable.

Also, with the introduction of immunohistochemistry and molecular pathology, the diagnostic work associated with each biopsy has become more complex. Due to an increasing number of biopsies, and the increasing complexity in specimen analysis, most pathology laboratories are in need of methods that make cancer diagnostics more efficient.

Histological grading of breast cancer tumors is important as it provides prognostic information. It can also influence treatment strategies (Helsedirektoratet, 2019). Histologic grading is therefore an important part of the routine diagnosis of breast cancer surgical specimen.

For the remainder of this chapter, we will give a brief technical background to digital pathology, breast cancer diagnostics and breast cancer diagnostics and histological grade.

8.1 Digital pathology

Most pathologists use brightfield microscopes to evaluate histopathological tumors sections. In recent years, digital pathology has become increasingly more popular, and it is currently being implemented in pathology laboratories nationwide. By scanning histopathological glass slides at high resolution, digital images are produced. These digitized sections can be examined on the computer screen instead of using a regular microscope.

There are several advantages with digital pathology:

1. Stored data do not lose quality over time
2. Digital annotations and measurements, i.e. tumor diameter
3. Teleconsultation
   - Easier consultation with colleagues locally
   - Easier consultation externally (no need to send histopathological sections and tumor blocks by mail)
4. Flexibility : can work from home
5. Sharing : easy and fast sharing of images with clinicians, students or multidisciplinary meetings
6. **Possible advantages of utilizing computational pathology → CADs**
   - Workflow more efficient
   - Low cost computational diagnostics
   - Increased consistency in diagnostics
   - Easy to design new and compare diagnostic markers

There are also some disadvantages and limitations associated with digital pathology:

1. Scanners are quite expensive and scanning time might be long
2. Requires a change in the current workflow

3. Not necessarily faster workflow

4. Loss in resolution in digitized slides, dependent on scanning configuration. Higher resolution requires overall longer scanning time

5. For computational pathology, requires additional expensive hardware to make diagnostics in real time

In the diagnostic setting, there is good concordance between regular and digital microscopy for a various different diagnostics (Rakha *et al.*, 2018, Bankhead *et al.*, 2017). In the study by Rakha *et al.* (2018), using 1675 samples, the best correlation of features relevant for histological grade was achieved at the highest magnification of 40x (Rakha *et al.*, 2018). Only mitoses were slightly effected during scanning. In addition, there was a study on concordance between digital WSI and standard glass-slide interpretation in teleconsultation. There was found a 91 % concordance of 53 cases, making teleconsultation a viable option (Wilbur *et al.*, 2009).

8.2 Breast cancer diagnostics and treatment

To ensure a correct breast cancer diagnosis, patients undergo a three-step examination: clinical examination, imaging, and preoperative biopsy of the lesion(s). The biopsy is formalin-fixed, and embedded in a paraffin block. Thin sections are made from the paraffin block. These sections are placed on a glass slide and stained with **haematoxylin and eosin** (HE). The sample is then examined by a pathologist.

HE is the most widely used histologic stain, due to its ability to contrast different tissue structures in a section. The haematoxylin component stains cell nuclei in a blue/black colour, and the eosin component stains the cytoplasm and connective tissue fibres pink, orange and red (Bancroft and Gamble, 1979).

Most breast cancer (BC) patients undergo surgery, removing the tumor and some surrounding tissue. Based on the biological properties of the tumor, patients may also be offered adjuvant treatment such as: anti-hormonal therapy, chemotherapy, targeted anti-HER2 treatment and radiation therapy.

Selecting the appropriate patients for chemotherapy remains a challenge. Patients expected to have a good prognosis after surgery should not be given chemotherapy, since treatment is associated with side effects both in the short and long term. Patients with a presumed poor prognosis need to be identified in order to be given suitable adjuvant treatment to improve their prognosis.

8.3 Histological grade

In 1991, Elston and Ellis defined a grading system, commonly known as **Nottingham grading system**. They proposed three key features/factors relevant for classification of breast cancer grade. These were: *tubular formation*, *nuclear pleomorphism*, and *mitotic activity*. Their scoring system gave a score for each subcategory, these were added to provide the **histological grade**. For each subcategory there are certain thresholds. The scoring system is briefly summarized in Table 8.1.

Table 8.1: Nottingham grading system summary

Factor	Criteria	Score	
Tubule formation	Majority of tumor, $>75\%$	1	
	Moderate degree, $[10, 75]\%$	2	
	Little or none, $<10\%$	3	
Nuclear pleomorphism	Very similar in size/shape, <1.5	1	
	Moderately larger, $(1.5, 2)$	2	
	Significantly larger, >2	3	
Mitotic count	Low count, ≤ 8	1	Dependent on field diameter (here: 5.6 mm)
	Moderate count, $(8, 17)$	2	
	Significant count, ≥ 18	3	

WHO's Classification of tumors of the breast (Lakhani *et al.*, 2012), provides guidelines for histological grading:

- Only tubules with a clear central lumen surrounded by polarized neoplastic cells, should be counted as tubules.

- For nuclear pleomorphism, the area with the worst nuclear morphological features should be assessed in the grading. Both size and variability in size should be taken into account.

- Mitotic counts/rate describes how many mitoses are present within ten high power fields (HPF) in the microscope. Counting should be done in the area with the *highest proliferative activity* (hotspot). This area is most often found in the periphery of the tumor. The area in one high power field varies from one microscope to another, and the cut-off levels for mitotic counts depend on the microscope field area. A more in-depth description of how the limits depends on field diameter can be found hereElston and Ellis (1991).

9 Data acquisition and pre-processing

In this chapter, we explain how we did the pre-processing for the CNN architectures, including tumor annotation extraction, tissue segmentation, and patch generation and data storage.

In this study, we used digital sections of a subset of tumors (n=62) from a well-described cohort of Norwegian breast cancer patients (Engstrøm *et al.*, 2013). The cohort comprises women invited to attend a clinical breast cancer screening study that took place between 1956-59 in Nord-Trøndelag, Norway (Kvåle *et al.*, 1987). The women were followed for BC occurence, and between 1961-2008, 1393 new BC cases were registered. All available tumors were reclassified into histological grade, and 909 tumors were reclassified into molecular subtypes (Engstrøm *et al.*, 2013).

Each sample was scanned at 40x magnification using Olympus scanner BX62VS with VSI120-S5, and stored in the cellsens vsi-format (default for Olympus). Each tumor was graded independently by two pathologists. In case of discrepancy, the sample was re-examined by two pathologists together, and consensus was made. In total, 62 WSIs were included (20 grade I, 21 grade II, 21 grade III). Only well-preserved tumor slides were included in this subset.

For the selected WSIs, tumor regions were annotated and controlled by two pathologists, using the open source software QuPath (Bankhead *et al.*, 2017). For each case, the data comprised a QuPath-project file, which included paths to all WSIs and respective annotations, stored in QuPath's qpdata-format.

In total, the raw data of the WSIs was 581 GB. A server was used for storing raw WSIs, as well as all processed data, and trained networks. To communicate with the server, we used a program called `PyCharm`.

In addition, we were later given an additional independent set of 40 WSIs for evaluation (8 grade I, 18 grade II, 14 grade III). These were randomly drawn from the full set of WSIs, and they were annotated by another pathologist.

9.1 Preprocessing

We designed a pipeline for processing WSI in `Python`. Using `Python` gives the developers more freedom, also making us able to use the well-established deep learning module `Keras`. This also makes it possible to run processes on GPUs, using `TensorFlow` backend.

9.1.1 Tumor annotation extraction

First, we would need to extract annotations from QuPath in a format that `Python` understands, i.e. binary masks. Since annotations were stored in the qpdata-format, only QuPath seemed to be able to read them. Due to memory restrictions inherit in QuPath, it was not possible to extract these directly. However, in QuPath, there is a built-in script editor (and compiler) which makes it possible to apply scripts on images. These scripts had to be written in `Groovy`, which is a high-level `Java` language that QuPath is built upon. In the script editor, there is also a *Run for project*-option, which made us able to run the same script on a collection of images, which automatized the process further.

It was possible to run QuPath from the command line, as well as include which project and image it should be initialized with. However, it was not possible to run scripts by it, since the functionality was not yet available. Thus, we were forced to run annotation extraction using the QuPath GUI. If we could run QuPath properly from the command line, we could also do the annotation extraction from `Python`, which would have automatized the processing further.

We propose to extract annotations as binary masks writing a `Groovy`-script.

As all qpdata-files were connected from *one* Project-file, we could apply the same script on all images within a project, using the built-in function *Run for project*. Then, for all WSIs respectively, we extracted tumor annotations and saved these as binary segmentation masks, in the PNG-format. Because of QuPath's inherit memory restriction, we could not save the full region. Thus, we found that simply downsampling the mask by an order of ten, made extraction possible.

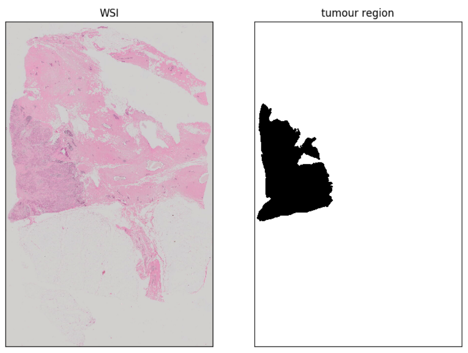


Figure 9.1: Comparison of WSI with mask

The order of ten was found empirically to be the lowest downsampling we could do on our data set. We did not want to downsample too much, since it would degrade the tumor boundary, but since annotations were done somewhat coarse, the overall smoothing should not be a problem. Also, for our purpose, it is not necessary to have a perfect tumor mask. Thus, the loss of resolution should not matter too much. To speed up extraction, each process ran in parallel using 16 threads, which only took a few seconds. All PNGs were stored in a single folder, containing WSI number, type of tissue and downsampling factor in the name.

We could not extract all tiles from the images, since there were so many redundant tiles, i.e. all glass, normal tissue, fat, etc. Hence, to make training more stable, speed up processing time, reduce storage usage, and speed up prediction time, only tiles from inside the tumor annotations were accepted. *Thus, we hypothesized that local information in the tumor was sufficient for histological grade prediction.*

9.2 Proposed solution

CNNs are quite computationally expensive. The images included in this study, can be as large as 200.000×160.000 pixels. This is too large to be sent directly through a CNN. One could downsample the images to 512×512, and thus be able to use traditional deep CNNs on high-end CPUs. However, for histological grading, two of the factors (nuclear pleomorphism and mitotic count) are studied at high magnification. Thus, such downsampling of the image would result in too much loss of relevant information.

A common solution is to divide the image into smaller patches (tiles). With this approach, global information is lost, since it is not currently computationally feasible to introduce global information, training a CNN-based patch wise classifier. It is however possible to train a classifier based on high magnification. Using the highest magnification level of 40x, only nuclei pleomorphism can be assessed. Mitotic counting is done in the hotspot area. Thus, grading based on mitotic counting in all patches does not make sense.

The degree of tubule formation is typically studied at much lower magnification levels by pathologists, i.e. 2x. However, it is possible to study it at higher resolution, but this results in reduced global information in a single patch. We found 10x to be the magnification level that produces the best trade-off between relevant factors. We did not have time to study more than one magnification level in the end.

9.3 Tissue segmentation

When using the tumor annotations as a mask for patch generation, some patches will be irrelevant for Breast Cancer (BC) grading, i.e. patches mostly containing glass and fatty tissue. It is therefore of interest to design a filter that can remove these components inside the tumor region, to avoid introducing too much noise in training, and to make prediction more robust.

We separated glass from tissue using the HSV color domain (Appendix B). The distinction was made based on the degree of dilution with white light: Glass is fully diluted with white light, while tissue contains mostly pure color. Hence, filtering based on the saturation channel, will provide a good segmentation, as seen in Figure 9.2 e).

To make thresholding more consistent, and to reduce the amount of fragments in the binary map, we propose to use a median filter prior to the thresholding. A kernel size of 7 was found empirically to be the best choice, but results were not extremely sensitive to the size. For thresholding, we propose to use Otsu's method (Appendix B), which was the same global thresholding technique used for lung segmentation in section 5.2. This method should work as long as there is some visible glass in the image. This should be the case for histopathological slides, as the tissue section will not fill the whole slide.

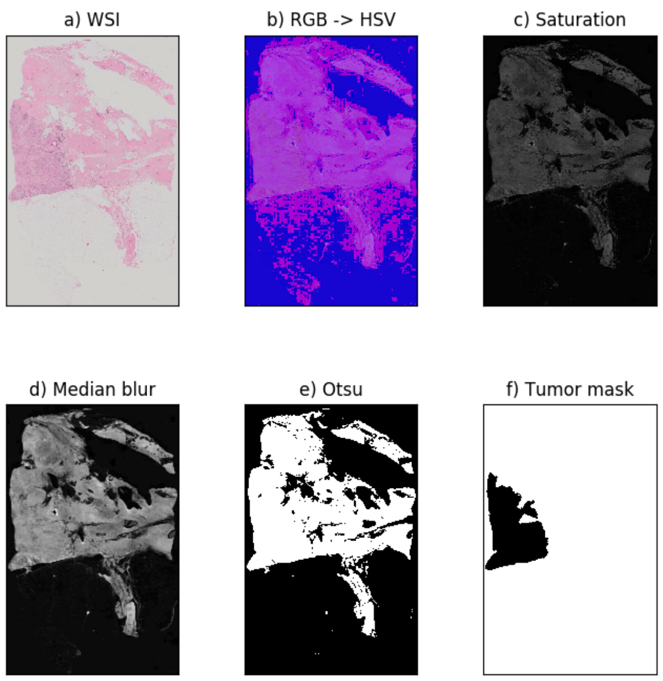


Figure 9.2: Illustration of how the tissue detector work. The final tissue mask can be seen in e). The corresponding tumor mask can be seen in f), and is only included for comparison.

9.4 Patch generation and data storage

We wanted to introduce as much global information as possible during training. The largest RGB image we could use with our CNN architecture was 512×512. Thus for all patches extracted, we used a fixed FOV or receptive field of this size.

To extract patches at different magnification levels, we took advantage of the structure of the vsi-format. The image is stored as an **image pyramid** (see Figure 9.3), which means that downsampled versions of the original image are stored. Data is not stored in the vsi-file, but in a corresponding frame.ets-file. The path to the frame.ets-file is located in the vsi-file, which is necessary to read the image. The original image can be found from extracting image information from the null image plane. For 10x magnification, one chooses the second image plane, and so forth. Without this image pyramid, zooming or extracting different image magnification levels, would not be feasible, due to the huge image sizes.

For all magnification levels, patches would have to be extracted from each vsi-file, and stored in a more suitable format for further processing. To read vsi-files in `Python`, the only available open source module is `Bioformats`, which is written in `Java`. However, in `Python`, there is a wrapper which makes us able to

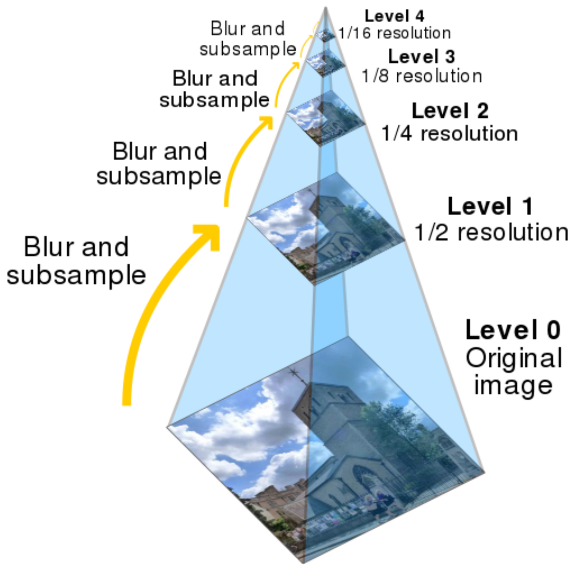


Figure 9.3: Visual representation of an image pyramid with five levels. Observe that for an increment in bit level, the image is downsampled by two.

use this Java-based module. In order to use `Java`-modules in `Python`, we would have to make a "java bridge", which was possible using the `Python-Javabridge` module.

It was still not possible to read the full image in memory - even on a supercomputer. Therefore, when generating patches from the image, we would only read from a pre-defined window. This was possible using the `Bioformats`-module. Because of memory and storage restrictions, we could not generate tiles from overlapping regions. Using non-overlapping patches also sped up processing, and made it possible to create a patch-generator which could be parallelized. Overlaps would produce patches containing the same information, which would result in synchronization problems, because of shared variables. All this is beyond the scope of this thesis, but quickly summarized, we could say that parallelization becomes more challenging, which results in some processes being forced to run sequentially - bringing parallelization down, which is exactly the opposite of what we try to accomplish.

We only extracted patches from inside the tumor annotations. To handle the boundary patches, only patches with more than 75% tumor annotations were accepted. Inside the tumor annotations there were also some irrelevant components - glass and fat. Hence, we used the generated tissue mask and removed all patches that did not contain more than 75 % tissue.

To reduce storage capacity, each patch was stored as a RGB uint8 type and compressed PNG format. Using the open source `Python` module `OpenCV`, it was memory efficient and fast. `OpenCV` is known to have the fastest `Python`-image reader/writer available.

NOTE: No additional pre-processing of tiles, i.e. normalization, color standardization, was done in this step. During training, we defined a new patch generator, which did additional pre-processing steps on-the-fly. In prediction, we only intensity-normalized each individual patch. All other augmentation used during training, should ideally not be necessary during prediction. A summary of the patch generation can be seen in Algorithm 2.

Algorithm 2 Summary of patch generation

```
   Input FOV, magnification level, location to all .vsi-files
   Output saved patches for each WSI in each respective folder
1: for each WSI do
2:     get corresponding grade label from WSI-file
3:     get corresponding tumor mask
4:     create folder with grade label, FOV and magnification level in the name
5:     initialize patch generator given list of WSI locations
6:     for patch from patchgen do                ▷ extract patches in parallel
7:         if patch > 75% tumor annotation & patch > 75% tissue then
8:             intensity normalize current patch
9:             save patch as RGB uint8 type, in the PNG-format, in current folder
```

10 Tumor segmentation

In this chapter, we will design and evaluate a 2D-UNet architecture for breast tumor segmentation from WSIs, experimenting with and without color augmentation.

To apply our method on new WSIs, the tumor region must be given since patches are only extracted from inside the tumor region. This makes the network dependent on the user (pathologist), and the overall workflow less automatic. We therefore aimed to use a simple machine learning method for automatic tumor segmentation, as a pre-processing step for the classifier.

Pathologists use both low and high magnification to segment the tumor, but a good initialization is done at the lower levels. Therefore, we propose that low-resolution information is sufficient for tumor segmentation. For our purpose, it does not require point-perfect tumor markings to be useful, as long as not too much redundant information is kept, and not too much tumor is lost.

Therefore, we downsampled all gigapixel resolution WSI-images to 512×512, and used these to train a UNet-architecture for tumor segmentation. Because of this harsh downsampling, our method is a naive approach.

10.1 Design

We used the same 2D-UNet architecture as for lung segmentation in section 5.1. If not mentioned otherwise, all other training and architecture choices were similar as for lung segmentation. The original network only handled grey level images, thus we needed to re-design the architecture to handle RGB input. In `Keras`, if the input has one more dimension than expected, it assumes that the last dimension corresponds to channels - as is the case with RGB images. This did not change any actual coding, but it is important to understand that there is a change in the resulting architecture.

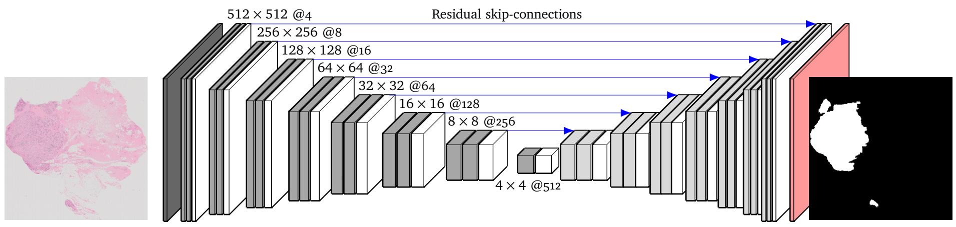


Figure 10.1: Illustration of the architecture chosen. The input is a RGB image of specified size in the first layer. The output is a softmax confidence map of two classes. 1 tumor, 0 background.

We also used the same soft dice loss function, which would not have been directly possible if we chose to do a patch wise approach. This is because there would be mostly tiles with all-tumor or all-background. Training autoencoders for these types of patches, does not really work. One would need to balance the classes in some way, either physically or with weights during training, but it is not trivial to do so since most of the patches will contain only a single class. Our approach does not require any balancing, since our loss function is only evaluated based on performance in segmenting tumor - performance on segmenting background is not interesting.

Due to our approach resulting in very little training data, we applied some data augmentation, but it is natural that our naive approach might not generalize well either way. We used random 90° rotations and flip. Initially we tried to use the traditional Macenko (Macenko *et al.*, 2009) and Vahadane (Vahadane *et al.*, 2015) H&E-stain-specific color augmentation, but they both were slow and seemed to degrade performance. At a late stage, light HSV augmentation seemed to work. Hence, we experimented with and without this technique.

To further evaluate the model, we have tested three other simple methods, which all tumor segmentation methods should outperform in order to be useful; tissue detector, guess all tumor, random guess. If a tumor segmentation method performs worse than a normal tissue detector (which detects all tissue, not only tumor), then clearly the method is not good/robust enough to be used in practice.

10.2 Evaluation

To evaluate the performance in segmentation, we used recall, precision and DSC. Dice score is the most common metric for overall segmentation performance. Recall was used to highlight performance in detection of tumors. Precision was used to evaluate performance in reduction of false positives. This is analog to how lung nodule detection and segmentation performance were evaluated in Chapter 7.

10.2.1 Monte Carlo cross-validation

To evaluate the method we used cross-validation (CV). It did not make sense to do three-split, as it seemed imperative to use a large test/validation set (18 WSIs, $\sim$ 40%) during training, to avoid us picking a model which had overfitted against the validation set. Doing simple two-split K-fold resulted in quite few estimates. Thus, we did two-split Monte Carlo CV with 10 iterations. Two thresholds were seen to give optimal DSC for the initial set; th=0.4 and th=0.6. Hence, we did CV using both of these.

For color augmentation, there was only time to do eight iterations in Monte Carlo CV. It should be noted that splits were random between color augmentation and without. Hence, test sets were not necessarily the same between these two methods. This unwanted artefact was not done on purpose, but rather observed at a late stage. Hence, there was not time to re-train the models. Results can be seen in Table 10.1.

From the produced trained models from CV, it was of interest to evaluate the method on the additional data set of 40 WSIs. This was done in a similar manner as above, but where the test sets were substituted with the new one. Results can be seen in Table 10.2. With this additional evaluation, both methods are evaluated on the same test set, making the comparison of the methods as fair as possible.

10.2.2 Bootstrapping

In practice, one typically only pick a single model to use for tumor segmentation, in the final design. It is possible to use multiple ones as an ensemble, but this was not further analysed. Hence, from the CVs, we picked the model that performed best on the initial data set for each design. We also did so for the same two optimal thresholds. Evaluating a single model on a single data set, results in one average estimate of each metric of interest. Hence, to say something about the uncertainty in the performance of the model, evaluated on both data sets, we used bootstrapping to find the BCa interval of each measured parameter.

Similarly as in section 10.2.1, we evaluated the resulting models on both data sets, with and without color augmentation, for the two same thresholds. Results from the initial and new data set can be seen in Table 10.3 and Table 10.4, respectively.

10.2.3 Experiments

We experimented with different color augmentation and prediction threshold designs to produce the optimal tumor segmentation mask. We also compared our proposed approach against three non-ideal approaches that all tumor segmentation methods should outperform to be useful. These were: tissue detector, guess all tumor, and random guess.

10.3 Results

10.3.1 Monte Carlo cross-validation

Table 10.1 shows the segmentation performance of the different approaches explained in section 10.2.3, after doing Monte Carlo CV, evaluated on the first data set. It seems like our UNet-design performs rather well in breast tumor segmentation, as DSC is the highest for all UNet designs compared to the other naive approaches. Note that DSC for the tumor detector is actually quite high, which means that approximately 74.4 % of the tissue in the average WSI in this data set is all tumor (assuming perfect tissue segmentation). However, predicting all tumor performs much worse, which is an indication that there is also a huge proportion of glass.

Nonetheless, the tissue detector is then essential for evaluating the performance in separating normal tissue from the tumor, which is best studied looking at recall and precision. The tissue detector produce almost perfect recall, which is because it says that all tissue is the tumor. All UNet designs perform worse in recall, but is much better suited for separating the tumor from other tissue types, as precision is much higher.

It is clear from the models, that setting a lower prediction threshold, increases recall, but reduces precision, and *vice versa*. The best performing model on the first data set was found using $th = 0.4$, without any color augmentation, according to DSC. Using color augmentation produced higher recall, but way lower precision.

Note that the tissue detector does not find all tumor-tissue, compared with the ground truth. Assuming a perfect tissue detector, we can argue that this is because some of the tumor-annotations include larger regions of fat/glass, which the tissue detector is not designed to include. This will probably also affect the trained models, as annotations of such structures were not consistent. However, the main contribution is probably because the method is not perfect, and tend to smooth boundaries where annotators have been more precise.

Comparing results from Table 11.1 and Table 11.2, it is interesting to see that performance is similar on both data sets, which indicate that the network has generalized well. However, note that the tissue detector performs better in terms of DSC on the new data set, as precision is much higher. This means that the trained models, which have similar DSC, probably performs slightly worse on the new data set, using the tissue detector as reference.

From this study, light HSV color augmentation seem to degrade segmentation performance. Using a lower prediction threshold produces better overall DSC. This is clear for both data sets.

Table 10.1: Results of doing monte carlo CV, where all trained models are evaluated against the same test set. Results are reported as: estimated sample mean (standard deviation)

Methods	Recall	Precision	DSC
2D-UNet$^{color}_{th=0.4}$	0.958 (0.054)	0.783 (0.197)	0.845 (0.153)
2D-UNet$^{color}_{th=0.6}$	0.949 (0.061)	0.797 (0.195)	0.849 (0.149)
2D-UNet$_{th=0.4}$	0.922 (0.150)	0.844 (0.179)	0.860 (0.153)
2D-UNet$_{th=0.6}$	0.880 (0.203)	0.866 (0.176)	0.841 (0.189)
Tissue detection	0.980 (0.024)	0.624 (0.184)	0.744 (0.150)
All tumor	1. (0.)	0.220 (0.100)	0.350 (0.131)
Random guess	0.500 (0.002)	0.220 (0.100)	0.293 (0.094)

Table 10.2: Results of doing monte carlo CV, where all trained models are evaluated using the new data set as test set. Results are reported as: estimated sample mean (standard deviation)

Methods	Recall	Precision	DSC
2D-UNet$^{color}_{th=0.4}$	0.957 (0.062)	0.786 (0.230)	0.836 (0.188)
2D-UNet$^{color}_{th=0.6}$	0.946 (0.069)	0.798 (0.230)	0.839 (0.186)
2D-UNet$_{th=0.4}$	0.956 (0.068)	0.804 (0.201)	0.854 (0.156)
2D-UNet$_{th=0.6}$	0.930 (0.116)	0.825 (0.197)	0.852 (0.159)
Tissue detection	0.969 (0.036)	0.709 (0.215)	0.795 (0.188)
All tumor	1. (0.)	0.281 (0.131)	0.423 (0.163)
Random guess	0.500 (0.001)	0.281 (0.131)	0.341 (0.115)

10.3.2 Bootstrapping

Table 10.3 and Table 10.4 show the performance on each data set, respectively, selecting the best performing model on the initial data set. Naturally performance should be better than for the results in Table 10.1 and Table 10.2. Now we are not evaluating the method, but rather the performance of *one* individual trained model.

It is interesting to observe that even though point estimates of the UNet-methods are higher than the other methods, they do not perform *significantly* better, for none of the data sets, as the intervals overlap for each respective metric. Looking at the intervals of all metrics, it is clear that they are quite skewed, which is an indication that it was a good choice to make BCa intervals, compared to the naive symmetric percentile intervals. This is also an indication that the method struggles more in some cases than others, producing outliers in evaluation, which means that its robustness is questionable.

It is interesting to see that the UNet-model actually performs better on the new data set, which is a good indication that the **one** trained model we have chosen generalizes well. However, as discussed earlier, the tissue detector also performs better.

Table 10.3: Performance of the best performing model on initial data set evaluated on the same data set. Results are reported as: estimated sample mean, confidence interval

Methods	Recall	Precision	DSC
2D-UNet$^{color}_{th=0.4}$	0.937 [0.911, 0.962]	0.868 [0.791, 0.925]	0.895 [0.855, 0.928]
2D-UNet$^{color}_{th=0.6}$	0.919 [0.886, 0.950]	0.884 [0.812, 0.936]	0.896 [0.863, 0.928]
2D-UNet$_{th=0.4}$	0.934 [0.877, 0.963]	0.863 [0.755, 0.916]	0.884 [0.780, 0.923]
2D-UNet$_{th=0.6}$	0.883 [0.799, 0.929]	0.891 [0.788, 0.939]	0.873 [0.812, 0.918]
Tissue detection	0.980 [0.963, 0.988]	0.624 [0.531, 0.701]	0.744 [0.658, 0.802]
All tumor	1. [NA]	0.220 [0.178, 0.271]	0.350 [0.292, 0.414]
Random guess	0.500 [0.499, 0.501]	0.220 [0.178, 0.273]	0.293 [0.250, 0.337]

Table 10.4: Performance of the best performing model on initial data set evaluated on the new data set. Results are reported as: estimated sample mean, confidence interval

Methods	Recall	Precision	DSC
2D-UNet$^{color}_{th=0.4}$	0.945 [0.921, 0.963]	0.799 [0.711, 0.857]	0.839 [0.763, 0.884]
2D-UNet$^{color}_{th=0.6}$	0.931 [0.903, 0.950]	0.813 [0.722, 0.872]	0.841 [0.766, 0.885]
2D-UNet$_{th=0.4}$	0.967 [0.948, 0.979]	0.818 [0.746, 0.863]	0.872 [0.814, 0.905]
2D-UNet$_{th=0.6}$	0.944 [0.917, 0.963]	0.850 [0.783, 0.891]	0.881 [0.829, 0.911]
Tissue detection	0.969 [0.955, 0.978]	0.709 [0.629, 0.766]	0.795 [0.723, 0.841]
All tumor	1. [NA]	0.281 [0.241, 0.322]	0.423 [0.369, 0.470]
Random guess	0.500 [0.499, 0.500]	0.281 [0.241, 0.322]	0.341 [0.301, 0.374]

11 Patch wise classification

In this chapter, we will design and evaluate a CNN-based classifier for histological grade prediction, both in the multiclass and binary (grades *I* and *III*) case, experimenting using different color augmentation techniques.

11.1 Design

To design a classifier for histological grade prediction we used a VGG-inspired architecture which was also used for malignancy prediction. However, in the current case we had to reduce the complexity in order to produce better generalization. We greatly reduced the number of convolutions in each layer, and also only used one single hidden dense layer prior to the output layer, using way less neurons than for the VGG16-architecture. The final architecture can be seen in Figure 11.1.

We ended up using extracted patches of size 512×512 from only a single magnification level of 10x, as we hypothesized that it was best suited for patch wise BC grading, and there was only time to study one. Similar to VGG16, we doubled the number of convolutions in each layer by two for each level, a total of five times, starting with four and ending up with 128. We used two convolution layers in each level. We only used a single hidden dense layer of 100 neurons, to reduce the complexity of the model. In the output layer, we used a softmax activation function of K outputs, dependent of how many classes we wanted to use for training and prediction.

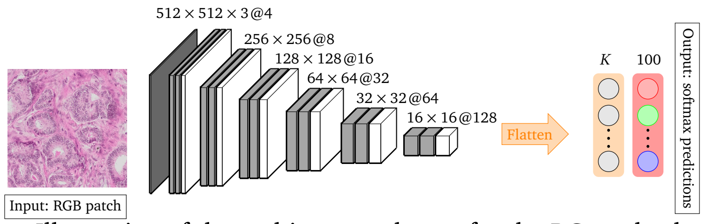


Figure 11.1: Illustration of the architecture chosen for the BC grade classification.

In the dense layers we also used dropout with rate 0.5, and in all convolution layers we used batch normalization. For optimization we used Adadelta with the same settings as previously. Categorical cross entropy was used as the loss function, as we one-hot encoded the labels. We monitored validation classification accuracy, and saved the best performing model based on this criterion.

Prior to training, WSIs were randomly split into three sets. Four WSI of each class were assigned to the test and validation set, respectively. The rest were used for training. Then for the test and validation set, we balanced the classes by random downsampling each class.

We designed a patch generator that processed a random patch from a defined set, using the image reader in `OpenCV`. Hence, it was necessary to convert from BGR to RGB, as `OpenCV` reads images as BGR. Then we RGB normalized each individual patch, before doing any further processing.

For data augmentation, we did random flip and 90°-rotations, and we experimented with HSV color augmentation (Appendix B). The best setup was to randomly add [-10,10] to the full hue and saturation channel, independently. For saturation we clipped the intensity bound if intensities reached outside the uint8-range after augmentation. For hue, we used modulus of 180, as hue was only defined between [0,180], and the intensity bound was defined as a "circle" in the HSV domain. Thus, the value 0 and 180 represent the same value, and this was the most natural way of handling the boundaries. For color augmentation, it was found that the network would only generalize if we only color augmented the training set. Hence, only flip and rotation was done on the validation set.

11.2 Evaluation

For evaluating all trained networks and designs we evaluate performance similarly as done for the malignancy classifier in Chapter 6. However, in this case we report patch wise and WSI level classification performance. In addition, we also included confusion matrices in the multiclass case. The reported summary performance table(s) should give the same type of information as the confusion matrix, but presented using relevant metrics like: precision, recall and F1-score.

We also include the macro average in addition to the weighted average for all metrics. As performance between classes varied a lot and samples were imbalanced, both of these contain different information about overall classification performance. Patch wise and WSI level classification performance was reported for the corresponding test set(s) in the original data set, for the new set only WSI level classification performance was reported.

We did *not* do any bootstrapping in either cases to generate confidence intervals for relevant estimates. For patch wise classification, patches were dependent on WSI. Thus, we could not directly bootstrap patches, as the bootstrapping assumption failed. In the WSI case the metrics reported was from a WSI level, thus independent. However, there was not enough variance in the samples. In the test set for the multiclass case a total of twelve values in $\{1, 2, 3\}$. This produced confidence intervals which ranged from [0, 1], or overall did not make sense. Also F1-score was dependent on both recall and precision. If both of these became zero, the F1-score would be ill-defined.

11.2.1 Experiments

In this study, we used a single magnification level of 10x, trained with the exact same architecture and training parameters, but for three different color augmentation setups: no color aug, light HSV (10), and heavy HSV (20). There was not time in the end to do a proper cross-validation for all setups. Therefore, for each color augmentation setup, we did three iterations of Monte Carlo CV and chose the single best-performing model on the test set. The same split was used for light and heavy HSV in the multiclass case.

For the three setups, we evaluated the trained models on each respective test set from the original data set (12 WSIs), and evaluated the models on the new set of 40 WSIs.

Finally, we tried to classify grades $\{I, III\}$, to see if it would generalize to the new 40 WSIs. We also experimented with four different levels of HSV augmentation: None, light (10), heavy (20), and heavier (30). In addition, we did a last experiment adding weights to the confidence maps of each grade, to see if it could produce better WSI classification accuracy. In the binary case the sample split was used for all designs as performance did not seem as dependent on the data split.

11.3 Results

11.3.1 Multiclass classification

From the confusion matrices of Table 11.1 (a), it is evident that the network performs best separating grades $\{I, III\}$, while it struggles the most separating grade II and III. The results are somewhat similar using color augmentation. However, for Heavy HSV, the model performs way worse.

From Table 11.2, light HSV produced best patch wise classification performance overall on the corresponding test set. However, we cannot say whether light HSV produces significantly better patch wise classification, since we were unable to make CIs or do hypothesis testing. Evaluation was also done on two different test sets. However, from the F1-scores, there is a tendency that the network perform better classifying grades I and III, compared to II. Using light HSV, it seems like the network is somewhat able to classify grade-II-patches. There is also a tendency that with color augmentation the network performs better classifying grade III than grade I. This conclusion differs from the case not using color augmentation.

From WSI classification (Table 11.3 and Table 11.4), using corresponding test sets, it is observed that even though patch wise accuracy is higher using light HSV, with the current consensus design, it performs worse than without color augmentation. For both none and light HSV, the network has detected all eight grade I and III-WSIs. There is also perfect classification for grades $\{I$, III$\}$. Without color augmentation, the network has only misclassified a single grade II-tumor as grade III. With light HSV, the method has misclassified twice the tumors of grade II as grade I. Heavy HSV fails to classify WSIs, as seen from the macro and weighted averages - performing similarly to random guess ($ACC_{random} = 1/3$).

Table 11.5 and Table 11.6 show the overall performance in WSI classification on the additional data set of 40 WSIs. It is clear from all HSV designs that the overall performance is much worse, compared to the initial study. Without color augmentation, the network ended up only guessing grade I. Using light HSV seemed to produce better transferability to the new data set, but still with a macro classification average of 0.351, it does not perform better than random guess. Surprisingly, best performance was achieved using heavy HSV.

Patch wise classification

Table 11.1: Confusion matrices for all three color augmentation setups, for patch wise classification

Class	1	2	3
1	2661	581	8
2	326	1297	1092
3	237	808	3161

(a) None

Class	1	2	3
1	2462	35	19
2	683	1064	482
3	99	426	5226

(b) Light HSV

Class	1	2	3
1	951	726	839
2	356	550	1323
3	829	2714	2208

(c) Heavy HSV

Table 11.2: Summarized patch wise classification performance for all three color augmentation setups

Class	Precision	Recall	F1-score	Support
1	0.825	0.819	0.822	3250
2	0.483	0.478	0.480	2715
3	0.742	0.752	0.747	4206
Macro avg	0.683	0.683	0.683	10171
Weighted avg	0.699	0.700	0.700	10171

(a) None

Class	Precision	Recall	F1-score	Support
1	0.759	0.979	0.855	2516
2	0.698	0.477	0.567	2229
3	0.913	0.909	0.911	5751
Macro avg	0.790	0.788	0.777	10496
Weighted avg	0.830	0.834	0.824	10496

(b) light HSV

Class	Precision	Recall	F1-score	Support
1	0.445	0.378	0.409	2516
2	0.138	0.247	0.177	2229
3	0.505	0.384	0.436	5751
Macro avg	0.363	0.353	0.353	10496
Weighted avg	0.413	0.353	0.375	10496

(c) Heavy HSV

WSI classification

Table 11.3: Confusion matrices for all three color augmentation setups, for WSI level classification

Class	1	2	3
1	4	0	0
2	0	3	1
3	0	0	4

(a) None

Class	1	2	3
1	4	0	0
2	2	2	0
3	0	0	4

(b) Light HSV

Class	1	2	3
1	2	0	2
2	1	1	2
3	0	3	1

(c) Heavy HSV

Table 11.4: Summarized WSI level classification performance for all three color augmentation setups

Class	Precision	Recall	F1-score	Support
1	1.000	1.000	1.000	4
2	1.000	0.750	0.857	4
3	0.800	1.000	0.889	4
Macro avg	0.933	0.917	0.915	12
Weighted avg	0.933	0.917	0.915	12

(a) None

Class	Precision	Recall	F1-score	Support
1	0.667	1.000	0.800	4
2	1.000	0.500	0.667	4
3	1.000	1.000	1.000	4
Macro avg	0.889	0.883	0.822	12
Weighted avg	0.889	0.883	0.822	12

(b) light HSV

Class	Precision	Recall	F1-score	Support
1	0.667	0.500	0.571	4
2	0.250	0.250	0.250	4
3	0.200	0.250	0.222	4
Macro avg	0.372	0.333	0.348	12
Weighted avg	0.372	0.333	0.348	12

(c) Heavy HSV

Evaluation on new data set

Table 11.5: Confusion matrices for all three color augmentation setups, for WSI level classification, evaluated on the new data set

Class	1	2	3
1	8	0	0
2	18	0	0
3	14	0	0

(a) None

Class	1	2	3
1	7	1	0
2	17	1	0
3	7	1	6

(b) Light HSV

Class	1	2	3
1	4	4	0
2	4	13	1
3	3	5	5

(c) Heavy HSV

Table 11.6: Summarized WSI level classification performance for all three color augmentation setups, evaluated on the new data set

Class	Precision	Recall	F1-score	Support
1	0.200	1.000	0.333	8
2	0.000	0.000	0.000	18
3	0.000	0.000	0.000	14
Macro avg	0.067	0.333	0.200	40
Weighted avg	0.040	0.200	0.067	40

(a) No color aug

Class	Precision	Recall	F1-score	Support
1	0.226	0.875	0.359	8
2	0.333	0.056	0.095	18
3	1.000	0.429	0.600	14
Macro avg	0.520	0.463	0.351	40
Weighted avg	0.545	0.350	0.325	40

(b) light HSV

Class	Precision	Recall	F1-score	Support
1	0.333	0.500	0.400	8
2	0.591	0.722	0.650	18
3	0.833	0.357	0.500	14
Macro avg	0.586	0.526	0.517	40
Weighted avg	0.624	0.550	0.548	40

(c) Heavy HSV

11.3.2 Binary classification

As seen in Table 11.7, for patch wise classification in the binary case, the same trend applies for different color augmentation techniques. Although the network does not perform as bad using heavier augmentation classifying only grade I and III. Still without grade II, the network performs worse using heavier HSV augmentations.

For WSI classification, the best performance is achieved with light HSV augmentation, also for grade $\{I, III\}$ binary classification. It achieved perfect classification, which makes sense as a network trained also including grade II, in the multiclass case, did the same for grades I and III, with light HSV.

It is interesting to see that although patch wise performance was worse for heavy and heavier HSV, at WSI-level, it only misclassified a grade I-tumor grade III, for both. Without color augmentation, the network misclassified a grade III-tumor as grade I once. Other than that, the performance of the network was perfect.

From evaluation on the new data set, without doing any color augmentation, the method simply fails, even though the problem is much simpler than the multiclass case. This is clear since the f1-scores is similar as for a random guess.

It is interesting to note that for the binary case, the network performs quite well classifying grades I and III, using light HSV. With perfect sensitivity of grade 1, and misclassifying *six* grade III-tumors as grade I. Doing heavier HSV also augmentation performed quite well, which was surprising, as it was expected that it would generate too many unnatural images which became irrelevant for histological grade prediction.

Patch wise classification

Table 11.7: Summarized patch wise binary classification performance of grades $\{I, III\}$, for all four color augmentation setups

Class	Precision	Recall	F1-score	Support
1	0.862	0.920	0.890	2751
3	0.947	0.907	0.927	4350
Macro avg	0.905	0.914	0.908	7101
Weighted avg	0.914	0.912	0.913	7101

(a) No color aug

Class	Precision	Recall	F1-score	Support
1	0.879	0.939	0.908	2751
3	0.960	0.918	0.939	4350
Macro avg	0.919	0.929	0.923	7101
Weighted avg	0.928	0.926	0.927	7101

(b) light HSV (10)

Class	Precision	Recall	F1-score	Support
1	0.804	0.756	0.780	2751
3	0.852	0.884	0.867	4350
Macro avg	0.828	0.820	0.824	7101
Weighted avg	0.833	0.834	0.833	7101

(c) Heavy HSV (20)

Class	Precision	Recall	F1-score	Support
1	0.661	0.593	0.625	2751
3	0.758	0.808	0.782	4350
Macro avg	0.710	0.700	0.704	7101
Weighted avg	0.721	0.724	0.721	7101

(d) Heavier HSV (30)

WSI classification

Table 11.8: Summarized patch wise classification performance of grades $\{I, III\}$ for all four color augmentation setups

Class	Precision	Recall	F1-score	Support
1	0.800	1.000	0.889	4
3	1.000	0.750	0.857	4
Macro avg	0.900	0.875	0.873	8
Weighted avg	0.900	0.875	0.873	8

(a) No color aug

Class	Precision	Recall	F1-score	Support
1	1.000	1.000	1.000	4
3	1.000	1.000	1.000	4
Macro avg	1.000	1.000	1.000	8
Weighted avg	1.000	1.000	1.000	8

(b) light HSV (10)

Class	Precision	Recall	F1-score	Support
1	1.000	0.750	0.857	4
3	0.800	1.000	0.889	4
Macro avg	0.900	0.875	0.873	8
Weighted avg	0.900	0.875	0.873	8

(c) Heavy HSV (20)

Class	Precision	Recall	F1-score	Support
1	1.000	0.750	0.857	4
3	0.800	1.000	0.889	4
Macro avg	0.900	0.875	0.873	8
Weighted avg	0.900	0.875	0.873	8

(d) Heavier HSV (30)

Evaluation on new data set

Table 11.9: Summarized WSI level binary classification performance of grades $\{I, III\}$, for all four color augmentation setups, evaluated on the new data set

Class	Precision	Recall	F1-score	Support
1	0.381	1.000	0.552	8
3	1.000	0.071	0.133	14
Macro avg	0.690	0.536	0.343	22
Weighted avg	0.775	0.409	0.285	22

(a) No color aug

Class	Precision	Recall	F1-score	Support
1	0.571	1.000	0.727	8
3	1.000	0.571	0.727	14
Macro avg	0.786	0.786	0.727	22
Weighted avg	0.844	0.727	0.727	22

(b) light HSV (10)

Class	Precision	Recall	F1-score	Support
1	0.389	0.875	0.538	8
3	0.750	0.214	0.333	14
Macro avg	0.569	0.545	0.436	22
Weighted avg	0.619	0.455	0.408	22

(c) Heavy HSV (20)

Class	Precision	Recall	F1-score	Support
1	0.500	0.875	0.636	8
3	0.875	0.500	0.636	14
Macro avg	0.688	0.688	0.636	22
Weighted avg	0.739	0.636	0.636	22

(d) Heavier HSV (30)

11.3.3 Visualization

For the pathologist using this CAD system, it would be beneficial to know why the classifier predicted the grade it did. For this reason we introduce heatmaps. From the sliding window predictions, the network outputs the probability of the patch belonging to each of the classes. This makes us able to create a confidence heatmap for each individual class. The predicted class of a patch is the one with the highest probability. Thus, we can make a prediction heatmap, illustrating the distribution of grades for the patches.

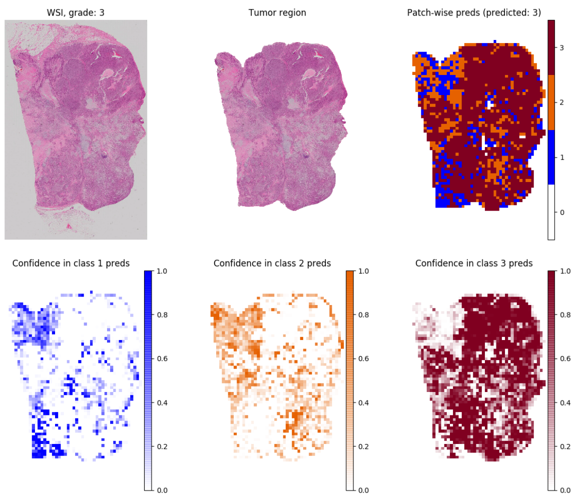

WSI, grade: 3
Tumor region
Patch-wise preds (predicted: 3)
3
2
1
0
Confidence in class 1 preds
Confidence in class 2 preds
Confidence in class 3 preds
1.0
0.8
0.6
0.4
0.2
0.0

Figure 11.2: Multiclass classifier

Figure 11.2 shows the produced heatmaps along with the original WSI and predicted tumor region in the multiclass case. Figure 11.3 shows the same but in the binary case. For the network, it was trained to only separate grade $\{RNum1$ and III. Hence, class two is not of interest, and there is no predictions for the intermediate class.

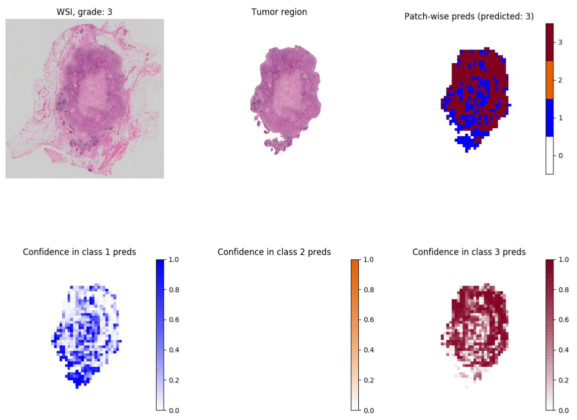

WSI, grade: 3
Tumor region
Patch-wise preds (predicted: 3)
3
2
1
0
Confidence in class 1 preds
Confidence in class 2 preds
Confidence in class 3 preds
1.0
0.8
0.6
0.4
0.2
0.0

Figure 11.3: Binary classifier

Ground truth class is given in the title of the original WSI, and the predicted histological grade is given in the title of the prediction heatmap.

Part III

Performance analysis and discussion

12 Concluding remarks

This chapter discusses the choices, designs and results of the different experiments for each subproject and also the outcome measured against the goals of the project. The first and second sections discussed the lung and breast cancer subprojects, respectively. The next section contains reflections on the use of deep learning to design cancer analysis CAD systems, as well as other topics including working with medical data, including traditional versus machine learning methods, cross-validation versus bootstrapping in evaluation, etc. Finally, there is a future work section, as well as a conclusion.

12.1 Lung cancer

12.1.1 Lung segmentation

In chapter 5, we trained a network for lung segmentation. It was trained on cropped lung-specific CTs, but failed to generalize to standard thoracic CTs, as seen in Figure 12.1. Most crucially, as seen in Figure 5.2, the network failed to include nodules in the mask. The traditional method did the same initially, but by using morphological post-processing these were included. This was also done to remove generated candidates outside the lung. Thus, the 2D-UNet works similarly to a regular intensity-based thresholder. In order to use the trained network as a lung mask in the reduction of FPR, one would need to do the exact same post-processing techniques already applied for the traditional method. Then, both methods should perform similarly, even on the LCTSC data set.

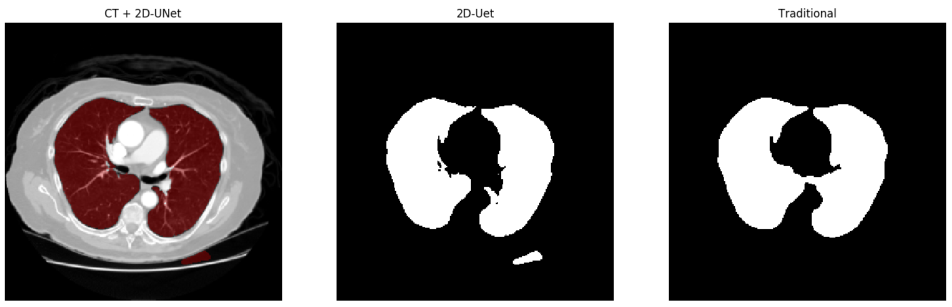


Figure 12.1: Comparing 2D-UNet against traditional intensity-based method evaluated on the LIDC data set.

Even though we did zoom-augmentation, to make the network more invariant to the size and location of the lung, it was not able to artificially generate the table. Hence, appropriate data is necessary in order to train a network for a specific problem.

The intensity based method can also be easily tuned to handle other cases, while the machine learning approach require sufficient amount of annotated data in order to generalize.

Note that even though we get strictly larger confidence intervals for the machine learning method, the trained network is evaluated on the validation set. The returned model is the one performing best on the validation set. Therefore, evaluation is not completely independent from training. We need an additional data set to evaluate which method performs best in lung segmentation.

We chose to use bootstrapping to get CIs for the estimated DSC. However, the estimates were not independent, as they were based on different trained models for each respective fold. Hence, the bootstrapping assumption failed, and the intervals are probably not accurate. Therefore, one should be cautious making too rash conclusions based on this study.

At a late stage, it was observed that there were made lung mask annotations for each patient in the LIDC data set, as part of the LUNA16-challenge (Setio *et al.*, 2017). These were made using a semi-automatic algorithm (van Rikxoort *et al.*, 2009). It was also stated in the challenge "*The lung segmentation images are not intended to be used as the reference standard for any segmentation study*". Therefore, these annotations would not be ideal for evaluating a lung segmentation method, as evaluation would be based on how similar the proposed method performs as to the semi-automatic method used to generate the ground truth, rather than comparing to the "actual golden standard".

12.1.2 Lung nodule-CAD performance

Lung mask

Using the naive lung mask without morphological post-processing produces a huge decrease in overall performance. It was expected that as it failed to include juxta-vascular nodules that sensitivity would drop. However, FPR was observed to increase. This happened because the lung mask applied did not remove all of the nodule. As some parts still remained, our not-too-strict object-wise sensitivity would still detect the nodule, if there was any overlap. But all other fragments would be counted as false positives. Hence, the FPR increases. This made sense looking at DSCTP in Figure 7.5 (b). If only a small fragment of the nodule was found, the segmentation performance would drop. Hence, it is imperative that a robust lung segmentation method that handles juxta-vascular nodules is used in the post-processing step in order to be useful.

Using the proposed lung mask did not greatly reduce FPR in the LIDC data set. Only a single nodule was generated outside the lung, for a prediction threshold of $th = 0.2$. This is an indication that the trained model does not often generate candidates outside the lung. Therefore, it is not necessary to apply a lung mask as a pre-processing step to train a lung nodule detector, which is what current state-of-the-art approaches typically do (Wu and Qian, 2019).

However, on the AAPM data set, there was much more to gain from using lung mask filtering. It was found that the CTs had more varying FOVs than the LIDC data set. Therefore, a similar artefact, as for the lung segmentation network, occurred. In one of our local data sets, it was found that the network had predicted that the right ear flip was a nodule. This made sense as the ear flip looked quite similar, as well as the surround area. This is an indication that the lung mask should be applied if the FOV differs from the standard thoracic CT, as the network is not trained to handle cases beyond this limit. However, it is not common to use larger FOVs for early stage lung nodule detection, but in many cases, lung nodules are found by accident in other diagnostics. Then the FOV might differ a lot, as well as imaging quality, depending on which organ is of interest. In future work, we will further evaluate the CAD system to which extent it handles such cases.

Note that even though the proposed lung mask did not degrade performance in the two data sets, it might still fail if there is varying imaging quality, or if regions in the lung has collapsed, or similarly. Patients greatly affected by emphysema and sarcoidosis have lungs looking way different than normal healthy lungs. In this case, traditional intensity-based methods fail to segment the lung. Thus, using the lung mask in this case, might result in filtering of actual clinically relevant nodules. Therefore, in future work we will either design an algorithm to detect emphysema or sarcoidosis, to evaluate whether or not to use the lung mask, or train a lung segmentation network that is invariant to these. Then it might be better to use a 3D-UNet for lung segmentation as the lung border might not look as smooth, making segmentation challenging in 2D.

Overlapping predictions

From the study in chapter 7, it was not certain whether overlapping predictions was beneficial or not for the CAD system, as it produced slightly worse performance on the LIDC data set, and slightly better performance on the AAPM data set. In future work, we will study different overlapping prediction designs to further boost performance. However, from this study, there was nothing to gain from using overlapping predictions.

It might be that since we are using way larger chunks than what is common, (common: $64 \times 64 \times 64$ mm^3, us: $256 \times 256 \times 64$ mm^3), there is sufficient global information for the network to detect nodules even located close to the border of the chunk. However, for smaller nodules, it is natural that the method will struggle either way, especially for lower longitudinal resolutions. Hence, overlapping predictions might be more beneficial if smaller nodules are relevant.

Evaluated on the AAPM data set, segmentation performance was better with overlapping predictions. This makes sense as we only keep the prediction volume from the central regions where the network is most confident. Hence, using overlapping predictions seem to be beneficial for segmentation performance, but it does not seem to be significant.

Filtering candidates on size

Filtering candidates on size makes sense as in the ground truth in LIDC, only candidates larger than 3 mm were annotated. At least in Norway, radiologists rarely study findings smaller than 5 mm further, as anything smaller is rarely malignant. Thus, it might be beneficial to filter candidates on size, as it might reduce FPR, making the job easier for radiologists.

On both data sets, it was beneficial to do filtering on predicted size, especially for the AAPM data set. Here, we found the setting $th = 0.6$ and $th_{size} = 4$, the best performance on the AAPm data set, giving average Recall = 0.840 and FPR = 1.1. On the LIDC data set, we achieve a Recall = 0.924 and FPR = 3.0, with the setting $th = 0.2$ and $th_{size} = 2$.

Note that the size thresholds used, as seen in the legends of Figure 7.4, does not correspond to mm. It was observed an implementation error in the pre-processing, resulting in voxels not being isotropic across patients. All CT-images were also downsampled by two. With an average transverse resolution of 0.7×0.7, the average voxel has the dimensions $1.4 \times 1.4 \times 1$, but will differ dependent on how much the transverse resolution differ for each patient. On average the actual nodule size will be approximately 1.96 higher, which makes sense, as sensitivity performance started to decrease around $th_{pred} = 2$. There was not time to correct for this mistake, but we will correct and re-train the model in future work.

Filtering ground truth on size

It was interesting to observe that the network only performed slightly better segmenting larger nodules. Hence, the trained model should be quite invariant to different sized nodules, which is important in order to be useful for filtering on predicted size. How it varies dependent on *nodule type*, would also be interesting to know, and will be evaluated in future work.

12.1.3 Malignancy prediction

For malignancy prediction, we tested five different CNN approaches, and surprisingly, there was no significant difference between them, evaluated on the LIDC data set. This is clear since all intervals of relevant metrics overlap. All designs produced quite good malignancy classification, which is probably because they were given good initializations during training.

For both translation augmentation designs, performance was way worse than without augmentation. Both translation designs produced networks that performed worse in recall on the benign class. This is probably due to benign nodules typically being quite small. Then doing translation produces initialization outside the nodule segment, making the problem much harder. This probably also explains why using heavier 3D-augmentation performed the worst, even including segments.

Adding segments did not improve performance, which is probably because initializations were good enough for the network to learn what was relevant (the nodule). Thus, even if it was slightly translated, the network would still be able to predict malignancy quite well, although the point estimate is somewhat lower.

A possible main drawback of study malignancy classification study, is that predictions can be based solely on the centroid. Thus, if the network under/over-segments, it does not matter, as long as we get a sufficiently good centroid. It may be beneficial to add predicted segments for handling off-centroids, when integrated with the nodule detector. However, it is not clear whether this is the case, from this study.

Unfortunately, there was not time to do an additional evaluation, using the AAPM data set or using the predicted segments/centroids, but this will surely be done in future work. Then it will be interesting to see if translation plays a bigger part than currently.

Intuitively, the local region around the nodule might be relevant for malignancy prediction, but according to this study, adding local background information did not improve classification performance, as seen in Figure 6.1 (d). However there was a slight drop in classification performance, but it was not significant. It is important to remember that the golden standard is set by the radiologists, and they only used nodule-features in the assessment of malignancy. Therefore, if we made a perfect malignancy classifier, it would only use nodule features, as it also seems like it does. In order to study whether adjacent information is relevant, we would need the "proper golden standard", which require a biopsy and a further analysis of the specimen by a pathologist.

We did not have time to do cross-validation as 3D-training was slow, and we tested many different designs. In future work, the method should be properly evaluated on the LIDC data set, using cross-validation.

12.1.4 Correlation between FPR and resolution

As already mentioned, training was not done with isotropic resolution, and in future work this will be corrected for. It is tempting to say that the reason for performance in FPR being low is due to this implementation error, but there are other reasons why the value we report might have been overestimated.

Figure 12.2 shows the distribution of the number of false positives after evaluation of the trained model on the 80 patients in the LIDC data set, using $th = 0.3$. The distribution of FPR seems rather skewed, with quite heavy tail. It is clear that three or four patients have much higher reported FPR than the rest. We found a strong correlation between FPR and resolution in the 80 patients. For all four "outlier"-patients, the longitudinal resolution was 2.5 mm or higher. For these cases, if there was a smaller nodule in the ground truth, it would only be noticeable in one or two CT slices. Then, since the resolution is low, we interpolate more, thus generating spheres from these "disks" from the original CT images.

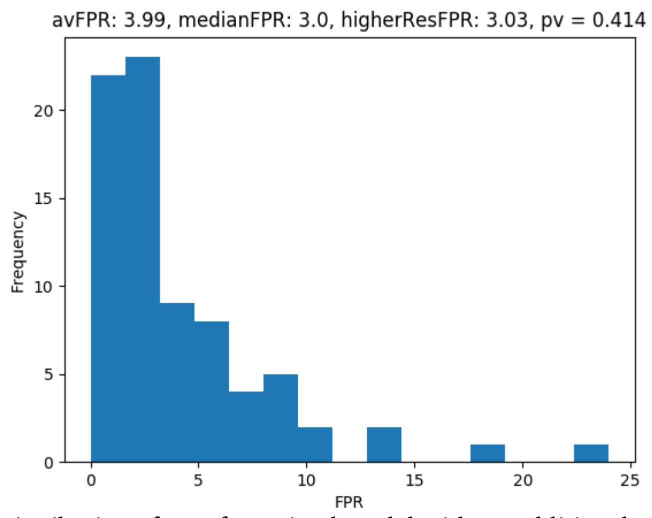


Figure 12.2: Distribution of FPR for trained model with no additional post-processing using prediction threshold $th = 0.3$. Reported FPR-measurements in title, as well as the estimated p-value for the hypothesis test done below

Thus we hypothesized that the median value for the "high-resolution"-sample $\tilde{\mu}_1$ was significantly higher than for the original sample $\tilde{\mu}_2$, and thus we did a Wilcoxons rank-sum test, with the following hypothesizes:

$$H_0 : \tilde{\mu}_1 = \tilde{\mu}_2 \qquad\qquad H_1 : \tilde{\mu}_1 > \tilde{\mu}_2 \tag{12.1}$$

This is a non-parametric test which only assumes that each each sample is drawn iid (independent and identically distributed), not necessarily from the same distribution, but that the distributions of each sample have *similar shape*. The p-value can be seen from Figure 12.2, and thus at a 5 % significance level we cannot say that the median FPR is significantly larger for the high resolution sample, compared to the original FPR sample.

However, it is interesting that the estimated mean FPR values are quite different, and it would make sense that FPR was lower for higher resolutions. This is because using longitudinal resolutions of 2.5-3 mm, produces "single-slice nodules", which are nodules only detected from one image in the CT. Since the resolution is quite poor, this might happen. However, it is extremely challenging to distinguish these smaller nodules from other structures and artefacts due to "insufficient" resolution. Thus, it is natural that the network struggles around this boundary.

Note that we also reported median FPR for the original sample in Figure 12.2. The median is not affected by outliers, and perhaps reporting median FPR might be a better choice than using the mean also for this case. However, the resolution in the median value is rather low. There is a huge difference between a FPR value of 2.5 and 3.4, but the median value does not distinguish between these. Thus, we reported the mean FPR instead.

12.1.5 LUNA design for FPR

In the LUNA-challenge (Setio *et al.*, 2017) they did a 75 % consensus design, as well as removing all patients with longitudinal resolution *larger* than 2.5 mm. The original ground truth from the LIDC data set, contain annotations of four radiologists, for both larger nodules, as well as markings for smaller ones. In the LUNA challenge they only counted false positives if the candidate was a *true false positive*. This implies that if one or two radiologists thought something was a nodule, but in the final 75%-design it was not, it would not be counted as a false positive. If any candidates overlapped with the smaller nodules, it would not count as a false positive either.

In our evaluation, this has not been done. We have been much more strict in evaluation. In addition we also trained *and evaluated* with longitudinal resolution larger than 2.5 mm, which means the the actual FPR we would get should be much lower than the ones we report.

In future work we will evaluate the model based on the LUNA-design, in order to be properly able to compare our approach against others.

12.1.6 Performance comparison

Note that we only evaluated the method using a single trained model on only the 80 first patients of the LIDC data. Therefore, one cannot really say that one method is better than the other. Not even the trained model give better detection and segmentation. We might have gotten lucky with the first 80 patients, and they might have reported the average point estimate from done K-fold CV.

However, doing cross-validation is not feasible for our design, as a single training of the produced model took 4-5 weeks. Therefore, in future work, we will evaluate the trained model on a new independent data set, instead of re-training multiple times. Then we will evaluate how well our trained network performs against other trained networks on the same data set.

In addition, more recent papers have adapted CPM, which is a metric based on the FROC curve, introduced in the LUNA-challenge Setio *et al.* (2017). Thus, it is uncertain whether any of these published papers perform better than our data set, as we have not calculated the same metrics, and thus cannot directly compare performance. In future work, we will also estimate CPM in order to be able to compare performance against these methods.

12.2 Breast cancer

12.2.1 Tissue detection

Of the HSV channels, we chose to use the saturation channel, as it was best suited for separating tissue from glass/fatty tissue. Results comparing different HSV channels can be seen in Figure 12.3. The original image can be seen in Figure 9.1. Using the naive grey channel, does not even include the tumor region, which is most important. This illustrates the importance of selecting a suitable color space/channel for this pre-processing step.

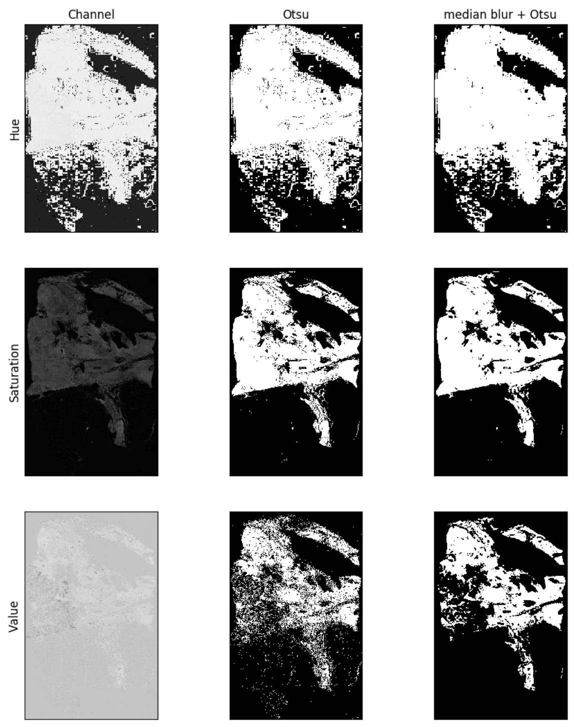


Figure 12.3: Tissue detection using different HSV channels.

Note that in the HE-color space, there is not any "green" color. Thus, it should be possible to separate breast tissue from glass, based on the green channel. The problem using the green channel is that this color space might not be applicable for other tissue types, using different stains. Hence, we chose to use saturation, as this concept should be more robust. Figure 12.4 illustrates the result of using any of the RGB-components for this processing step, and it is clear that both the green and blue color channel would be suitable, for this specific tissue type and staining. Here the effect of doing median blurring prior to the thresholding is much clearer, with fewer fragments, especially for the blue channel. Doing this additional step, produces outputs of similar quality. From the red channel, it is evident why the grey channel was a poor choice, since the red channel would highly influence the variation in intensities of the grey channel.

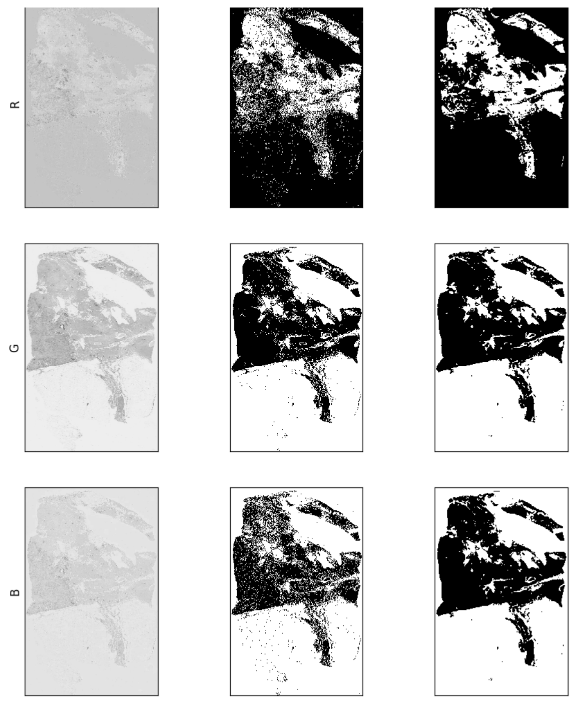


Figure 12.4: Tissue detection using different RGB channels.

When we do blurring prior to the thresholding, we remove some smaller glass components in the mask. It is not a clear answer whether one should do this or not. An argument to why one should, is because smaller glass components inside the tumor, might correspond to inside tubules, which is extremely relevant for histological grade predictions. By doing smoothing, these smaller patches would still be kept. However, it might be that these patches are actually almost only glass, and does not contain any tubules tissue at all. In this case one should remove the patch, as it becomes redundant. We chose to blur as losing a few patches inside the tumor region should not effect the overall WSI classification.

12.2.2 Tumor segmentation

Figure 12.5 shows the segmentation performance of the best model evaluated on the original test set. The network seem to be able to remove a lot of non-tumor tissue for all WSIs. For row 2, some homogeneous stroma regions have not been included in the ground truth, and the network fails to handle this case. By doing such heavy downsampling, the network struggles separating stroma from within and outside the tumor.

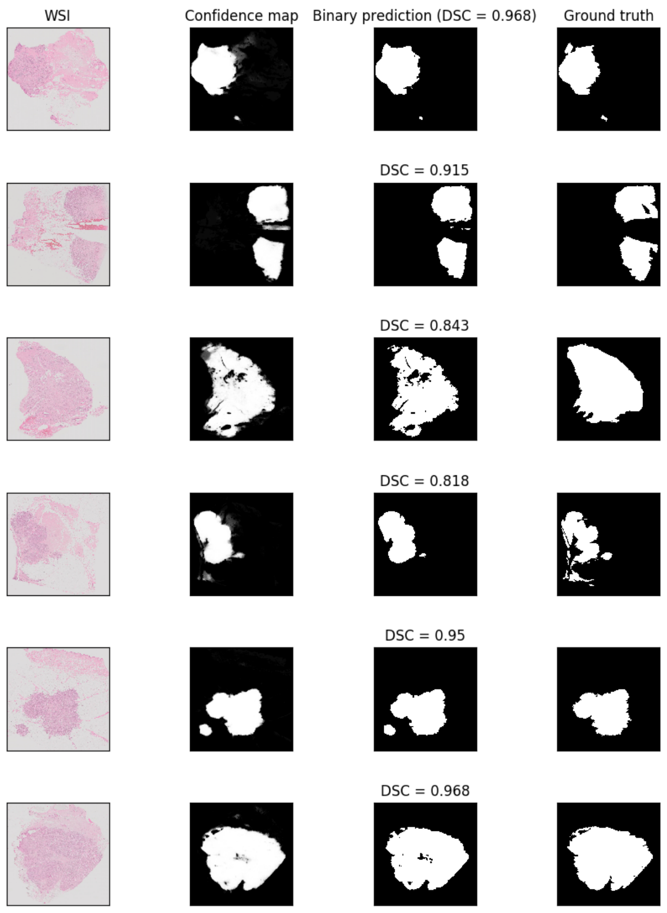


Figure 12.5: Segmentation performance of best network

In row 3, the network removes larger glass regions in the tumor, while in the ground truth these are included. This is perhaps an indication that these larger glass regions should not be included in the ground truth. It is possible to apply the tissue detector to handle this case.

Row 4 shows a case were the network struggles to segment the fine tumor boundary, effectively producing something similar to a blob. If we worked with higher resolution, the contour would have probably been better. Row 5 illustrates an interesting case were the network is able to detect a part of the tumor that was lost in the annotation/extraction of the ground truth.

Lastly, row 6 shows a case were the network is uncertain whether to remove the homogeneous stroma region in the middle of the tumor or not. This is not done consistently in annotations, as seen from the ground truths in row 6 and row 2. Therefore, the network also struggles.

Stroma problem

The biggest drawback from this study is that by doing heavy downsampling, the network is not able to distinguish between normal and tumor-stroma. Even the pathologist fails to be consistent in the annotations, as seen in Figure 12.5 row 2 and row 6. Nonetheless, given perfect annotations the network would still struggle with stroma, and this is why in future work more complex tumor segmentation methods should be tested, also accounting for high-resolution information.

Color augmentation

Using light HSV seemed to degrade performance. This makes sense as when we downsample the images as much as we do, color becomes an important factor. It is already tough enough to separate health and tumor stroma. By doing color augmentation we further smooth away this contrast, making the problem much harder for the network.

12.2.3 Histological grade classification

First of all, it was observed a (critical) flaw with the approach when doing evaluation. Prior to training we balanced patches to have the same number of patches for each class, both for the training and validation set. However, we did not balance the number of patches from each tumor. Tumors might vary a lot in size. Therefore, by not doing this balancing, smaller tumors will be under-represented in both the training and validation set.

Hence, the network would most likely base its decisions hugely influenced on how it performed on patches from the larger tumors, and less on how it performed on the smaller tumors. Also patch wise evaluation was done on a patch level, not taking this into account. It would be better to report patch wise accuracy for each individual WSI, and report the macro average across these. The patch wise weighted average patch wise accuracy we report, biases larger tumors in evaluation, as they are over-represented. In future work we will correct for this mistake.

Another problem with this, is that we cannot use bootstrapping to make BCa intervals for relevant measures, as patches are dependent of WSI, which means that the bootstrapping assumption fails. If we reported patch wise accuracy for each individual WSI, then the bootstrapping assumption would not have failed, and we could have created CIs.

Note that in the multiclass case, we found another split to produce much better overall classification performance on all three sets. However, the same did not apply for the light and heavy HSV. Therefore, we chose to evaluate performance on two different test sets, with and without color augmentation. Ideally, one would evaluate the model on the exact same test set, as well as do a proper cross-validation. However, there was not time to do this in the end. Nonetheless, we would still correct for the patch-tumor-size-problem assessed earlier before doing any evaluation first. In the binary case, we used the exact same data split, as performance did not seem as dependent on the data split.

Multiclass case

As to why the network performs better classifying I is because it is also the easiest one for the pathologist. Heavy HSV augmentation degraded overall performance. This is most likely due to heavier color augmentation generating unnatural images. It is also possible that the network had been dependent on color features which it should not have. As the data set is quite small, we cannot say for certain what the reason is. If heavy HSV actually removed some important information, it would generate more redundant patches. This would surely degrade performance.

Heavy HSV fails to classify WSIs, as seen from the macro and weighted averages - performing similarly to random guess ($ACC_{random} = 1/3$). This illustrates the importance of creating a solid patch wise classifier, before continuing to the next stage, as further predictions will be strongly influenced by what it has learned in the first stage.

The most probable cause as to why all designs failed on the new data set, is because the initial data set were effectively simpler than the additional data set. WSIs having the least amount of artefacts, degradations and overall simpler classification job was chosen. Thus, the network has not learned to handle more complex WSIs. As to why the tumor segmentation method did not get influenced as much, is probably because of the strong downsampling in the pre-processing step, reducing the effect of some artefacts. A lot of degradation in tissue is also best studied at higher resolution. As the method only uses extremely low resolution information, it was not influenced by this.

12.3 Deep learning based CADs

In this thesis, we have developed two deep learning-based CADs to be used for lung and breast cancer diagnostics. Results from the lung cancer-subproject produced close to state-of-the-art performance both in detection and segmentation on the benchmark LIDC data set. For the breast cancer-subproject, there has never been reported performance on BC-grading for all three grades. Performance was especially good for the initial data set, but failed to generalize to new data. Nonetheless, this study illustrates the power and benefits of using deep learning in the field of CADs. Deep learning-based methods are easily adapted between modalities, cancer types, problems and data types.

12.3.1 Segmentation

In this thesis, we designed a state-of-the-art autoencoder-design inspired by the UNet-architecture, to produce robust segmentation performance in lung, lung nodule and breast tumor segmentation. The same networks could be easily adapted to different problems, data sets and data types. Even on small data sets, the methods produced great results, on both lung and breast tumor segmentation, and seemed to generalize well, especially clearly on breast tumor segmentation.

12.3.2 Classification

For classification, we used a VGG16-inspired architecture for malignancy prediction of lung nodules and histopathological grade prediction of breast tumors. Both models performed extremely well on the initial data sets, but it is uncertain how well the models will perform on new data. Smart data augmentation, i.e. color augmentation, seemed to be provide better generalization, but it requires additional studies on new data sets to evaluate this further.

12.3.3 Local vs open data sets

Using open data sets, one does not always get the ideal data for training a network. An example of this was lung segmentation using the LCTSC data set. Even though we trained a network that outperformed the traditional intensity-based methods on the current data set, it did not generalize as well on data sets of different FOVs, due to lack of proper data (thoracic data which includes nodules in lung mask). Hence, in these cases it might be beneficial to use traditional image processing/machine vision approaches, as we observed for lung segmentation.

12.3.4 Machine learning vs traditional methods

Throughout this thesis, we have used machine learning methods for classification and segmentation, but in many cases, traditional methods are more suitable, as they might be more robust, requiring no training, and also be memory efficient and fast. An example is lung segmentation. Even though we could use a machine learning method for this problem, using a simple intensity-based method provided more robust performance on data sets of different FOVs. It is typically easier to tackle new scenarios when we define the algorithm ourselves. It is tougher to train a network to be invariant to changes it has not yet seen. Thus, for insufficient data, small data sets or lack of variance, traditional methods might be a better solution than machine learning methods. As well as pre and post-processing, traditional methods can always be useful. This is seen in tissue and lung segmentation, and post-processing of lung nodules.

12.3.5 Speed

Using a single GPU during prediction, we were able to make pipelines that processed from the raw data to give an output in a matter of seconds. The slowest process we made was the sliding window prediction in BC grading, but even here the slowest process was around 2 minutes. For smaller tumors, processing time was as low as 2-4 seconds. Total processing time from the raw cellsens vsi-format to predictions with heatmaps, took about 4 seconds - 2 minutes, mostly dependent on the tumor size. Hence, processing time is feasible for pathologists, but require further advances in classification performance to be useful in practice.

Lung segmentation using the 2D-UNet approach took less than a second on the GPU, while the traditional methods took about 6-7 seconds, on the CPU. For lung nodule segmentation, sliding window predictions without overlap took approximately two minutes on the CPU, while on the GPU took approximately 10 seconds. The overall pipeline took approximately 20-25 seconds to process one full CT-scan, processing from the raw DICOM-format. Hence, processing time is feasible for radiologists.

12.3.6 Data augmentation

For small data sets, data augmentation proved to be extremely beneficial for producing more robust and generalizable networks. This was especially clear in patch wise classification of histological grade. However, doing too strong augmentation, might degrade overall performance, as seen in the malignancy classification with heavy 3D translations, as well as with heavy/heavier HSV in patch wise classification. When generating artificial data, it is important that the new data represent the true population - in the sense that it is natural. A result of using heavier HSV color augmentation can be seen in Figure 12.5, producing unnatural staining.

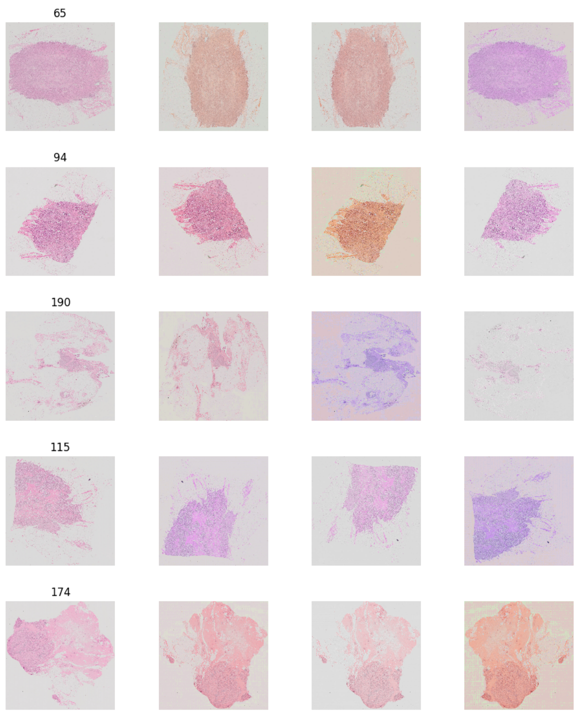
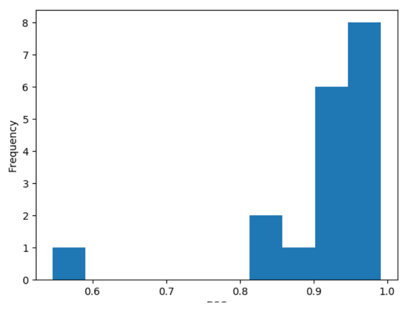


Figure 12.6: Too heavy HSV color augmentation produces unnatural stains

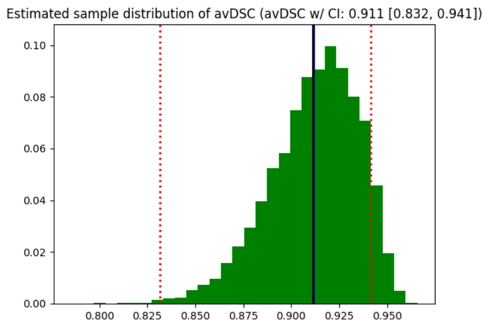
12.3.7 Bootstrapping

Even though bootstrapping is a fast and easy way to generate CIs, there are many cases were doing bootstrapping might not be the best idea. Even though one may assume that the measurements are drawn iid from the same distribution, it becomes problematic when the data set is small, there is not much variance in the data, and if there are outliers.

Figure 12.7 (a) shows the distributions of estimated DSC for breast tumor segmentation for patients in the test set. Here, most DSCs are around 0.9, but



(a) DSC



(b) avDSC

Figure 12.7: Comparing distribution of DSC and avDSC found using bootstrapping

then there is a clear outlier to the left close to DSC=0.5. The distributions is clearly not symmetric (left-skewed). Therefore, to estimate the BCa interval of avDSC, we used bootstrapping, which resulted in Figure 12.7 (b). Because of this serious outlier, doing bootstrapping might produce replicates of this value, which overall results in a bootstrapped avDSC that might be way lower then the average DSC of the original sample. Perhaps there is not enough variance in the data set to say whether or not this is an actual outlier. However, after doing bootstrapping, calculating the BCa interval was the right choice as the sample distribution of avDSC was quite asymmetric. Using simple percentile intervals her might produce an interval that even goes beyond the maximum DSC of 1.

12.4 Conclusion

Based on the work presented in this thesis, we draw the follow conclusions:

- We were successful in developing a multipotent deep learning-based CAD design which adapted well across two widely different imaging modalities and data types

- We have created two platforms for further development (breast) and testing (lung). These platforms may be used for further research by MSc and PhD students

- We were successful in developing a pipeline for processing WSI from the cellsens vsi format. The pipeline can be use in further research within the field of digital pathology

- For both subprojects, we were able to design CAD systems using deep learning, one of which, to the best of our knowledge, produced state-of-the-art performance in overall lung nodule detection and segmentation trained and evaluated on the LIDC data set

- Processing time from raw DICOM format is a few seconds. Processing time from raw vsi format is heavily depended on tumor size. The slowest reported processing time for vsi was two minutes

- For lung nodules we showed that the results can be easily visualized in CustusX, along with the predicted lung mask. For histological grade predictions we produced heatmaps to help pathologists interpret the results

12.5 Future work

This thesis has raised numerous questions which should be investigated further.

Future work in lung cancer diagnostics includes:

- Study augmentation techniques to make network more invariant to different resolutions and interpolation artefacts

- Train network on thoracic CTs to perform sufficient lung segmentation, to be used as a post-processing step

- Evaluate how well the studied malignancy classifier designs integrate with the lung nodule detector in the produced CAD system
- Train networks using raw CT data, as well as predicted segments and malignancy predictions, to make patient-wise predictions of lung cancer, also introducing patient characteristics
- Do a clinical study of the use of the Lung-nodule-CAD system prototype to further minimize workload and maximize efficiency in early detection of lung nodules

In breast cancer diagnostics, future work includes the following:

- Use more data to train and evaluate networks, and find which design choices are most relevant for each respective task
- Further study and develop new color augmentation techniques, compare these with color standardization, and study which are most suitable for different problems in breast cancer histopathology
- Correct for tumor-patch-imbalance in pre-processing to study whether it improves classification performance
- Study new network designs for tumor segmentation, as well as train a network to be invariant to different types of organs - using other augmentation techniques
- Study a two-step CNN classifier, first training a patch wise classifier using one CNN, then using the produced heatmap as input to a second CNN to predict histological grade
- Further introduce patient characteristics, as well as survival data, to study whether one is able to train CNNs to further divide grade the clinically heterogeneous grade *II* into two prognostic subgroups, then use CNNs and explainable AI to find new diagnostic markers
- Make a multi-tissue/structure segmentation network to be used as a pre-processing step to guide the patch classifier and reduce redundant number of patches
- Use immunohistochemistry for annotation of different tissues in sections in order to train multi-tissue/structure segmentation networks
- Produce a fully functioning prototype to be tested in collaboration with

the pathology department, as an assisting diagnostic tool

Finally, for CAD-systems, future work includes the following:

- Further develop CAD systems to be tested in collaboration with St. Olavs and NTNU, either as decision support or in surgery planning
- Introduce more data in training and evaluation, from different scanners with different scanning configurations, to produce more robust CAD systems
- Develop CAD systems that can make actions dependent on risk analysis, based on multi modal analysis and big data

A
Evaluation Metrics

This appendix gives a brief introduction to relevant metrics and methods for evaluating methods for classification and segmentation, as well as some image processing concepts relevant for pre-processing and segmentation of images.

First we will introduce some basic terms. Lets say we want to evaluate a binary classifier, for the classes A and B. Let G_A: ground truth is A, and G_B: ground truth is B. Let P and N denote the number of G_A (positive samples) and G_B (negative samples), respectively. The possible outcomes of the classification (or segmentation) is then summarized as:

(a) True positive (TP) : correct classification, $A \cap G_A$

(b) True negative (TN) : correct classification, $B \cap G_B$

(c) False positive (FP) : misclassification, $A \cap G_B$

(d) False negative (FN) : misclassification, $B \cap G_A$

FN is often referred to as type 1 error and FP type 2 error. This means that the we have the following null hypothesis H_0: GT is A. Lets say that A correspond to the tumor class and B the non-tumor class. A type 1 error is to reject a true null hypothesis, which would be to classify an actual tumor as a non-tumor. A type 2 is to fail to reject a false null hypothesis, which would be to classify a non-tumor as tumor. Typically, we try to minimize the type 1 error, as it is more critical to misclassify a tumor as a non-tumor, then the opposite.

It is common to summarize these terms in a **Confusion matrix**, which in the binary classification problem can be seen in Figure A.1.

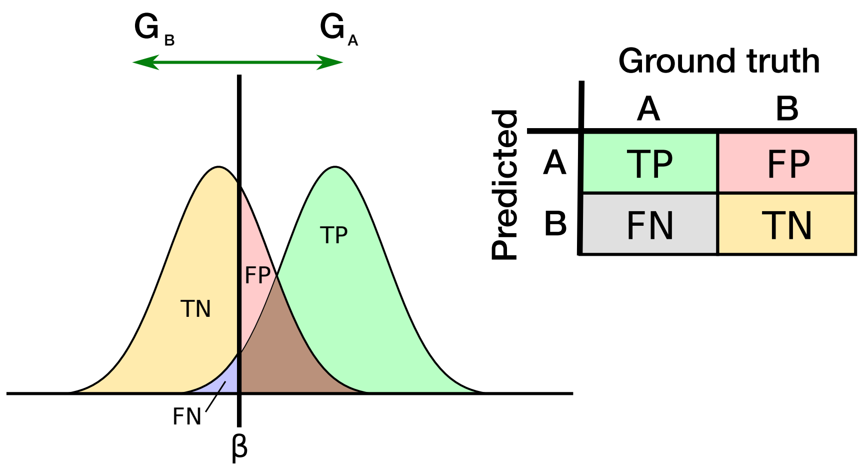


Figure A.1: Left: Illustration of the probability distributions related to the different classification results: Right: Illustration of the confusion matrix in the binary case. β correspond to some classifier/model parameter, i.e. prediction threshold in a trained neural network

a) Sensitivity (SN), also called recall (REC) or True Positive Rate (TPR), is the number of correct positive predictions divided by total number of positives, and is given by:

$$SN = P(A \mid G_A) = \frac{P(A \cap G_A)}{P(G_A)} \overset{estim.}{\simeq} \frac{TP}{TP + FN} = \frac{TP}{P} \tag{A.1}$$

b) Specificity (SP), also called True Negative Rate (TNR), is the number of negative predictions divided by total number of negatives, and is given by:

$$SP = P(B \mid G_B) = \frac{P(B \cap G_B)}{P(G_B)} \overset{estim.}{\simeq} \frac{TN}{TN + FP} = \frac{TN}{N} \tag{A.2}$$

c) Precision (PR), also called Positive Predictive Value (PPV), is the number of correct positive predictions divided by total number of positive predictions, and is given by:

$$PR = P(G_A \mid A) = \frac{P(G_A \cap A)}{P(A)} \overset{estim.}{\simeq} \frac{TP}{TP + FP} \tag{A.3}$$

d) Accuracy (ACC) is a common measure to evaluate classification. It is the overall number of correct predictions divided by total number of samples, and is given by:

$$ACC = P(\text{correct classification}) = P(A \cap G_A) + P(B \cap G_B)$$
$$\overset{estim.}{\simeq} \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{P + N} \tag{A.4}$$

e) F1-score is also a common measure to evaluate classification, but is known to handle imbalanced data sets better than ACC. It is defined as the harmonic average of precision and recall, and is given by:

$$F_1 = 2 \cdot \frac{\text{PR} \cdot \text{SN}}{\text{PR} + \text{SN}} \tag{A.5}$$

f) Dice Similarity Coefficient (DSC), also commonly referred to as Dice Score, is commonly used to evaluate segmentation performance. It is actually the same metric as F1-score, but is interpreted differently in segmentation evaluation, involving sets of values, not just a single one (predicted segment vs. ground truth segment). In the binary case, it is defined as the measured overlap between the predicted set S_P and the ground truth set S_G, and is then commonly given by:

$$DSC = \frac{2|S_G \cap S_P|}{|S_G| \cup |S_P|} \overset{bool}{=} \frac{2TP}{2TP + FP + FN} \tag{A.6}$$

where $|S_P|$ and $|S_G|$ correspond to the number of elements (or cardinalities) in each set. The reason for 2 in the nominator, is because of the double count of the overlapping volumes in the denominator. For boolean data it can be rewritten in terms of TP, FP and FN.

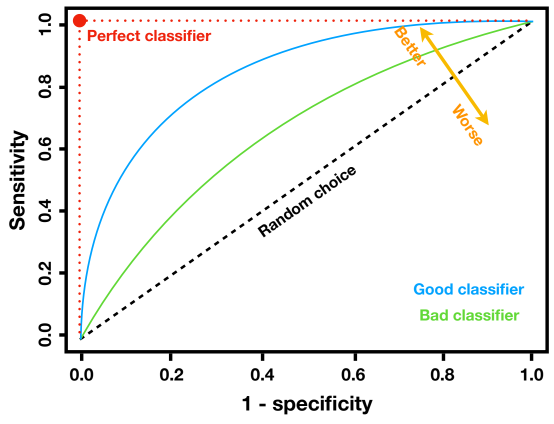


Figure A.2: Illustration of the ROC curve.

g) Receiver operating characteristic (ROC) curve, is one of the most common ways to visualize performance of classifiers. It shows True Positive Rate (sensitivity) as a function of False Positive Rate (inverse specificity). As seen in Figure A.2, the ROC-curve illustrates how the performance of the model, in terms of sensitivity and inverse specificity, varies as a function of some model parameter β (Figure A.1). The diagonal represent random choice performance. Thus, all models with performance closer to the top left, perform better than random choice.

h) Free-response Receiver Operating Characteristic (FROC) curve, is an alternative to the ROC curve. Instead of using FPR along the horizontal axis, it is used measure of FP. This is useful since FN is not always possible to measure in terms of detection. The difference then being, there is no diagonal representing random choice, as for ROC curve, but all other concepts still apply for FROC, as for ROC.

/B Image Processing

B.1 Colorspaces

The most well-known color model is RGB, which is based on the concept that "all" colors are subsets of the three primary colors: Red, Green and Blue. To generate a color, we find each color component, and concatenate to yield a triplet.

Each channel is assumed to be in the range [0,1], which means that the color model can be described as a uni-sized cube. For instance the tuples (0,0,0) and (1,1,1) corresponds to black and white, respectively. Red, green and blue have the tuples (0,0,1), (0,1,0) and (1,0,0). Other colors can be found by changing the values inside the tuples, i.e. bright orange (1, 0.62, 0.15).

B.1.1 HSI

Although RGB is popular, it is not well suited for describing colors in terms of human interpretation (Gonzalez and Woods, 2010b). When looking at the color of a house, humans do not think of the percentage of R, G and B-components, or that each color is a concatenation of three distinct color channels.

Humans rather see colors in terms of Hue, Saturation and Intensity (HSI, also called HSV - V for Value). Hue describes the amount of pure color (yellow, orange, red). Saturation is the level of white light diluted in pure color. Intensity is the grey level image, which can be found from taking the average of the RGB-components. It explains the level of brightness or luminescence in the image.

Conversion between HSI and RGB is easy to do, but not trivial to derive. It can be showed that the conversion transform is:

$$\begin{cases} H = \begin{cases} \theta & \text{if } B \leq G \\ 360 - \theta & \text{if } B > G \end{cases} \\ S = 1 - \frac{3}{R+G+B}[\min(R, G, B)] \\ I = \frac{1}{3}(R + G + B) \end{cases} \tag{B.1}$$

where $\theta = cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-B)^2+(R-B)(G-B)]^{1/2}}\right\}$. To go from HSI to RGB, one could just rewrite the equations above.

B.2 Thresholding

One of the most fundamental image processing operations is called **thresholding**. The idea is to split the intensities of the image $f(x, y)$ into two chunks based on a user-defined threshold k. All intensities higher than k is set to 1, and all other set to 0, effectively binarizing or segmenting the image. Mathematically this is denoted as:

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > k \\ 0 & \text{if } f(x, y) \leq k \end{cases}$$

where $g(x, y)$ correspond to the segmented output. By setting multiple thresholds it is possible to further segment the image into more segments or classes.

B.2.1 Otsu's Method

Finding which threshold is optimal for segmentation is tedious and often result in suboptimal generalization. Therefore, methods to generate *global* threshold(s), based on intensity information is explored. One of the most popular ones is called **Otsu's method** (Otsu, 1979). The idea is to find the threshold which maximizes the between-class variance and minimizes the within-class variance, similarly to what is done in Fisher Discriminant Analysis.

The Otsu's algorithm can be summarized as follows (Gonzalez and Woods, 2010c):

1. Compute normalized histogram of input image $f(x, y)$, where each histogram component is denoted as $p_i, i = 0, 1, ..., L-1$, where L is defined from the histogram width

2. Compute cumulative sums, $P_1(k), k = 0, 1, ..., L-1$

3. Compute cumulative means, $m(k), k = 0, 1, ...L-1$

4. Compute global intensity mean, m_G

5. Compute between-class variance, $\sigma_B^2(k), k = 0, 1, ..., L-1$

6. Obtain Otsu's threshold from which σ_B^2 is maximized. If value is not unique, find optimal threshold from averaging across all candidates

Further information about each term in the algorithm can be found here Gonzalez and Woods (2010c), Otsu (1979). The method can also be trivially extended to include n thresholds, but this will not be used in this thesis, and is therefore not included.

B.3 Convolution

A way of filtering an image $I(x, y)$, is to design a specific kernel function $K(x, y)$, which is applied in a sliding window fashion across the image. To apply the kernel K of size $m \times n$ on the $M \times N$ image I, we use the convolution operator, which typically is denoted with an asterisk:

$$S(x, y) = (I * K)(x, y) = \sum_{x=1}^{m} \sum_{y=1}^{n} I(x-m, y-n) \cdot K(m, n) \tag{B.2}$$

As the kernel is sliding across the image, the kernel is applied locally for each pixel, and thus updating the current centre pixel of the kernel.

To handle the border, it is common to *pad* the image with zeroes around the border (often referred to as **zeropadding**). This way, the convolution operator can be applied also on these pixels. Because of this padding, the output from the convolution function typically returns some artificial border, depending of how similar the border pixels are to zero and the kernel function used. If you do not pad, these pixels are excluded in the convolution, and a smaller image is returned - depending on the kernel size. This is referred to as **unpadded convolution**.

In pre-processing, it is common to use convolution to reduce noise in an image. An example is salt-and-pepper noise, which occur as small bright and dark pixels in the image. A natural way of removing these, is to apply a kernel function that returns the median value from the convolution kernel. The median is commonly used to remove outlier values, and thus is ideal for this type of noise. This filter type is called the **median filter**.

B.4 Mathematical Morphology

A way of extracting or segmenting objects of interest from images, can be done using (mathematical) morphology. The idea is to apply simple mathematical operations on the image, based on morphology, which filters the image based on which operator is chosen. Figure B.1 illustrates different operators applied on the same binary image.

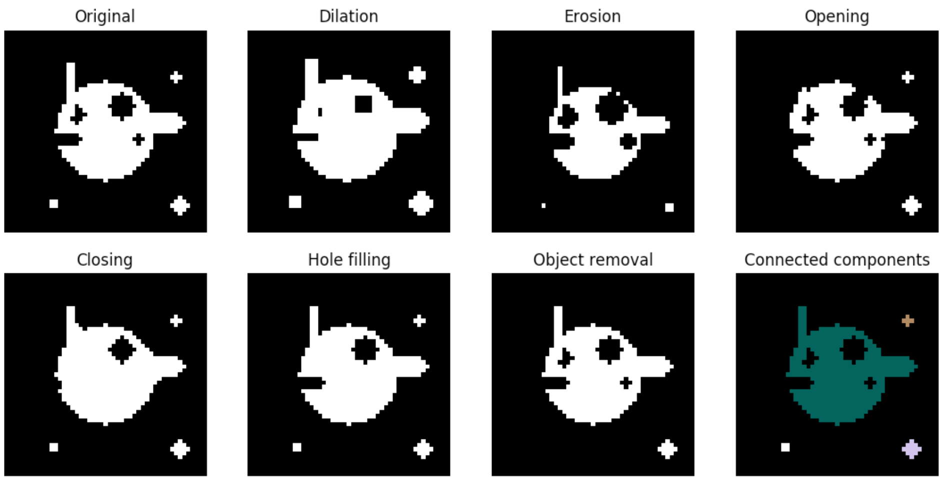


Figure B.1: Illustration showing the effect of using different types of morphological operators on the same image

Note that morphology can be applied also on any kind of image. The only constraint is on the **structuring element** chosen. For the image above a disk of varying size for each image was chosen to best illustrate the difference between the operators. Changing the size of the disk or using a different kernel might produce different results. For instance if some unwanted holes remain after using hole filling, applying a larger area threshold might help. Further information on the topic can be found here Gonzalez and Woods (2010d).

C Inference

C.1 Cross-validation

When evaluating a classifier, it is important to validate it based on an independent data set. Thus given that we have one data set available, it is common to do a *two-split*, where data is divided into two independent sets; train and test set. That way evaluation will be independent of training, and we get an estimate of how well the classifier will perform on new data.

The problem doing a single split, is that the classifier is only evaluated on a smaller subset of the full data. The estimate might also not represent the true estimate, if the test set was overall simpler than the training set, essentially introducing a bias. Thus, it is common to do **cross-validation**(CV), which is essentially re-splitting data multiple times and evaluating the classifier on the whole data.

The most common design is **K-fold CV**. Data is split into K chunks, or *folds*, of approximately equal size, where a single fold is used for evaluation and the remaining $K-1$ folds is used for training. By doing this training-evaluation K times on K different splits, we get K point estimates of the relevant parameter(s) $\hat{\theta}$, i.e. classification accuracy. Then a confidence/prediction interval can be found using these estimates.

Using neural networks, it is necessary to monitor performance on an independent data set also *during* training, to avoid the risk of overfitting (see section

2.4). If we used the test set for this monitoring, evaluation on the test set afterwards would not be completely independent from training. Especially not if we chose the model which produced the lowest error on the monitored data set during training. Thus, it is common to do a *three-split* of the data set, where we introduce the validation set, or simply val set.

Another way to do CV is to split the data in three, and then do K-fold CV for only train and val set. Then one trains the network K times, and evaluate it on the same test set. This should give a better estimate of how well the neural network will perform training on new data. Then evaluation of the K models will be evaluated on the same test set, which using neural networks will be completely independent from training, which is not the case for two-split CV.

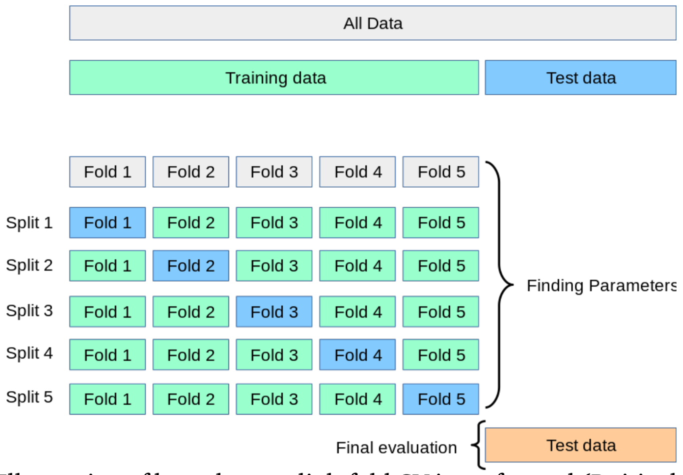


Figure C.1: Illustration of how three-split k-fold CV is performed (Buitinck *et al.*, 2013).

How to choose the splits are dependent on the number of samples available. There is no point in doing CV if the data set is sufficiently large, since then evaluation based on a single test set, should be close to the "true" value. In the case of smaller data sets, CV is important to reduce the effect of the split, since performance might vary more between test sets, than for larger data sets. For smaller data sets, it is common to divide the data set into 60/20/20 for train/val/test, respectively.

C.2 Bootstrapping

Given some observed data set, it might be of interest to estimate some parameter θ. Using the data directly it is possible to estimate a point estimate $\hat{\theta}$. What is commonly of interest is to estimate a confidence/prediction interval of the estimated value, in order to include information about uncertainty. This seems difficult to obtain, given that we only have a single value for the point estimate for the current data set. However, there exists statistical resampling techniques, which makes us able to extract this information. One popular resampling technique is **bootstrapping** (Efron, 1979).

It is possible to use non-parametric bootstrapping to estimate the sampling distribution of θ. It is based on the simple concept that one can create bootstrap samples by resampling (with replacement) the original data set. The resampling is done B times, where each bootstrap sample has the same length as the original data set.

This gives B bootstrap point estimates of $\hat{\theta}$. That way using bootstrapping you have *pulled yourself up by one's bootstrap*. Instead of getting more estimates directly, you have generated new ones solely based on the data set you have available.

To use bootstrapping the only assumption one makes is that the observed data $\boldsymbol{x}$ is drawn **iid** from some common population distribution F, such that:

$$\boldsymbol{x} = (x_1, ..., x_n) \overset{iid}{\sim} F(\theta) \tag{C.1}$$

That way if we were to actually gather more data to get more estimates, we should asymptotically approach the same conclusion under this assumption. Note that bootstrapping is only an approximate method to estimate the sampling distribution of the estimate $\hat{\theta}$. To improve the approximation one can increase the number of bootstrap samples generated. A common number of bootstrap samples is $B = 10000$.

C.2.1 Accelerated bias-corrected percentile method

In practice, it is quite common to assume some distribution on $\hat{\theta}$, e.g. a normal distribution or another symmetric parameter distribution. However, such a parametric distribution assumption is often not valid. A common way to calculate confidence intervals based on bootstrap samples is to use the **accelerated bias-corrected percentile method (BCa)** (Efron, 1987).

The BCa interval for θ is defined as:

$$P(\xi^*_{\gamma_1} \leq \theta \leq \xi^*_{\gamma_2}) \approx 1 - \alpha \tag{C.2}$$

where:

$$\gamma_1 = \Phi\left(b + \frac{b + z_{\alpha/2}}{1 - a(b + z_{\alpha/2})}\right)$$
$$\gamma_2 = \Phi\left(b + \frac{b + z_{1-\alpha/2}}{1 - a(b + z_{1-\alpha/2})}\right)$$

and where:

$\Phi(.)$: cumulative distribution function for the standard normal density
z_α : Quantile of the standard normal density
ξ^*_γ : Empirical quantile from the distribution of $\hat{\theta}$

Note that * marks estimations from bootstrap samples. The terms a and b correspond to the acceleration and bias-correction terms. A common suggestion is to define these as:

$$b = \Phi^{-1}\left(\hat{F}^*(\hat{\theta})\right) = \Phi^{-1}\left(\frac{\#\{\hat{\theta}^*(b) < \hat{\theta}\}}{B}\right)$$
$$a = \frac{\frac{1}{6}\sum_{i=1}^{n}\phi_i^3}{\left(\sum_{i=1}^{n}\phi_i^2\right)^{3/2}}$$

For $b = 0.5$, there will be no bias correction. For $b > 0.5$, it correct rightwards, and $b < 0.5$ leftwards.

b : Measures the median bias between between $\hat{\theta}^*$ and $\hat{\theta}$, and corrects estimate if there is a bias
a : Acceleration term. Refers to rate of change of the standard error of $\hat{\theta}$ with respect to the true estimate θ

ϕ_i correspond to the jackknife estimates, which can be found using:

$$\phi_i = \hat{\theta}_{(.)} - \hat{\theta}_{-i},\ i = 1, \dots, n \qquad \hat{\theta}_{(.)} = \frac{1}{n}\sum_{i=1}^{n}\hat{\theta}_i \tag{C.3}$$

This is actually a resampling technique called **jackknifing** (Efron, 1982), which is another way to estimate the population distribution. The idea is to apply a leave-one-out procedure, where $\hat{\theta}$ is estimated on the original data set excluding one value $\hat{\theta}_i$. This is therefore only done n times (McIntosh, 2016). Using bootstrapping one is able to introduce more variance in the resampled data sets. Therefore, it is more often used.

Bibliography

Aresta, G., Araújo, T., Kwok, S., Chennamsetty, S. S., P., M. S. K., Varghese, A., Marami, B., Prastawa, M., Chan, M., Donovan, M. J., Fernandez, G., Zeineh, J., Kohl, M., Walz, C., Ludwig, F., Braunewell, S., Baust, M., Vu, Q. D., To, M. N. N., Kim, E., Kwak, J. T., Galal, S., Sanchez-Freire, V., Brancati, N., Frucci, M., Riccio, D., Wang, Y., Sun, L., Ma, K., Fang, J., Koné, I., Boulmane, L., Campilho, A., Eloy, C., Polónia, A., and Aguiar, P. BACH: grand challenge on breast cancer histology images. *CoRR*, abs/1808.04277, 2018.

Armato, S. G., Hadjiiski, L. M., Tourassi, G. D., Drukker, K., Giger, M. L., Li, F., Redmond, G., Farahani, K., Kirby, J. S., and Clarke, L. P. Special section guest editorial: Lungx challenge for computerized lung nodule classification: reflections and lessons learned. *Journal of Medical Imaging*, 2(2):1 – 5 – 5, 2015.

Armato III, S., Mclennan, G., Bidaut, L., McNitt-Gray, M., R. Meyer, C., P. Reeves, A., Zhao, B., Aberle, D., I. Henschke, C., Hoffman, E., Kazerooni, E., Macmahon, H., Beek, E., Yankelevitz, D., Biancardi, A., H. Bland, P., S. Brown, M., M. Engelmann, R., E. Laderach, G., and Clarke, L. The lung image database consortium (lidc) and image database resource initiative (idri): A completed reference database of lung nodules on ct scans. *Medical Physics*, 38:915–931, 2011.

Armato III, S., Mclennan, G., Bidaut, L., McNitt-Gray, M., R. Meyer, C., P. Reeves, A., Zhao, B., Aberle, D., I. Henschke, C., Hoffman, E., Kazerooni, E., Macmahon, H., Beek, E., Yankelevitz, D., Biancardi, A., H. Bland, P., S. Brown, M., M. Engelmann, R., E. Laderach, G., and Clarke, L. Data from lidc-idri. the cancer imaging archive. 2015.

Armato III, S. G., Hadjiiski, L., Tourassi, G. D., Drukker, K., Giger, M. L., Li, F., Redmond, G., Farahani, K., Kirby, J. S., and Clarke, L. P. Spie-aapm-nci lung nodule classification challenge dataset. the cancer imaging archive. 2015.

Askeland, C., Solberg, O., Bakeng, J. B., Reinertsen, I., Tangen, G., Hofstad, E., Iversen, D., Våpenstad, C., Selbekk, T., Langø, T., Hernes, T., Leira, H., Unsgård, G., and Lindseth, F. Custusx: an open-source research platform for image-guided therapy. *Int J CARS*, 11:1–15, 2015.

Bancroft, J. and Gamble, M. Theory and practice of histological techniques. *Elsevier Sci.*, pages 85–88, 1979.

Bankhead, P., Loughrey, M. B., Fernández, J., Dombrowski, Y., McArt, D. G., Dunne, P. D., McQuaid, S., Gray, R. T., Murray, L. J., Coleman, H. G., James, J. A., Salto-Tellez, M., and Hamilton, P. W. Qupath: Open source software for digital pathology image analysis. *Scientific Reports*, 7(1):16878, 2017.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

Callister, M. E. J., Baldwin, D. R., Akram, A. R., Barnard, S., Cane, P., Draffan, J., Franks, K., Gleeson, F., Graham, R., Malhotra, P., Prokop, M., Rodger, K., Subesinghe, M., Waller, D., Woolhouse, I., and . British thoracic society guidelines for the investigation and management of pulmonary nodules: accredited by nice. *Thorax*, 70(Suppl 2):ii1–ii54, 2015.

Efron, B. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7, 1979.

Efron, B. The jackknife, the bootstrap and other resampling plans. society for industrial and applied mathematics. *Philadelphia*, pages 1–11, 1982.

Efron, B. Better bootstrap confidence intervals. *Journal of the American Statistical Association*, 82:171–185, 1987.

Elston, C. and Ellis, I. Pathological prognostic factors in breast cancer. i. the value of histological grade in breast cancer: Experience from a large study with long-term follow-up. *Histopathology*, 41:154–161, 1991.

Engstrøm, M., Opdahl, S., Hagen, A., Romundstad, P., Akslen, L., Haugen, O., Vatten, L., and Bofin, A. Molecular subtypes, histopathological grade and survival in a historic cohort of breast cancer patients. *Breast cancer research and treatment*, 140, 2013.

Fischer, P., Dosovitskiy, A., and Brox, T. Descriptor matching with convolutional neural networks: a comparison to sift. 2014.

Frossard, D. Vgg in tensorflow; model and pre-trained parameters for vgg16 in tensorflow, 2016. [Online; accessed May 12, 2019].

Golan, R. Deepcade: A deep learning architecture for the detection of lung nodules in ct scans. 2018.

Gonzalez, R. C. and Woods, R. E. Digital image processing 3rd edition. pages 31–35, 2010.

Gonzalez, R. C. and Woods, R. E. Digital image processing 3rd edition. pages 418–435, 2010.

Gonzalez, R. C. and Woods, R. E. Digital image processing 3rd edition. pages 760–769, 2010.

Gonzalez, R. C. and Woods, R. E. Digital image processing 3rd edition. pages 649–702, 2010.

Goodfellow, I., Bengio, Y., and Courville, A. Deep learning. pages 321–330, 2016.

Guérin, J., Gibaru, O., Thiery, S., and Nyiri, E. CNN features are also great at unsupervised classification. *CoRR*, abs/1707.01700, 2017.

Han, D., Heuvelmans, M., Vliegenthart, R., Rook, M., D. Dorrius, M., and Oudkerk, M. An update on the european lung cancer screening trials and comparison of lung cancer screening recommendations in europe. *Journal of Thoracic Imaging*, 34:1, 2018.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

Helsedirektoratet. Nasjonale anbefalinger, råd og pakkeforløp, 2019. [Online; accessed May 16, 2019].

Hohberger, L. A., Schroeder, D. R., Bartholmai, B. J., Yang, P., Wendt, C. H., Bitterman, P. B., Larsson, O., and Limper, A. H. Correlation of regional emphysema and lung cancer: a lung tissue research consortium-based study. *J Thorac Oncol*, 9(5):639–645, 2014.

Hojjat, S. and Kittler, J. Region growing: A new approach. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 7:1079–84, 1998.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

Jiang, H., Ma, H., Qian, W., Gao, M., and Li, Y. An automatic detection system of lung nodule based on multigroup patch-based deep learning network. *IEEE Journal of Biomedical and Health Informatics*, 22(4):1227–1237, 2018.

Kanashiki, M., Tomizawa, T., Yamaguchi, I., Kurishima, K., Hizawa, N., Ishikawa, H., Kagohashi, K., and Satoh, H. Volume doubling time of lung cancers detected in a chest radiograph mass screening program: Comparison with CT screening. *Oncol Lett*, 4(3):513–516, 2012.

Koyuncu, H. Lupsix: A cascade framework for lung parenchyma segmentation in axial ct images. *International Journal of Intelligent Systems and Applications in Engineering*, 6(4):322–328, 2018.

Krogh, A. and Hertz, J. A. A simple weight decay can improve generalization. In Moody, J. E., Hanson, S. J., and Lippmann, R. P., editors, *Advances in Neural Information Processing Systems 4*, pages 950–957. Morgan-Kaufmann, 1992.

Kvåle, G., Heuch, I., and Eide, G. E. A prospective study of reproductive factors and breast bancer: I. parity. *American Journal of Epidemiology*, 126(5):831–841, 1987.

Lakhani, S., Ellis, I., Schnitt, S., Tan, P., and van de Vijver, M. WHO classification of tumors. 4, 2012.

Latinovic, N. The significant difference between artificial intelligence, machine learning and deep learning, 2018. [Online; accessed May 12, 2019].

LeNail. Nn-svg: Publication-ready neural network architecture schematics, 2019.

Leung, A. and Smithuis, R. Solitary pulmonary nodule: benign versus malignant differentiation with ct and pet-ct. 2007.

Li, X., Chen, S., Hu, X., and Yang, J. Understanding the disharmony between dropout and batch normalization by variance shift. *CoRR*, abs/1801.05134, 2018.

Li, Q. Recent progress in computer-aided diagnosis of lung nodules on thin-section ct. *Computerized medical imaging and graphics : the official journal of the Computerized Medical Imaging Society*, 31:248–57, 2007.

Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

Loussaief, S. and Abdelkrim, A. Deep learning vs. bag of features in machine learning for image classification. In *2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, pages 6–10, 2018.

Macenko, M., Niethammer, M., Marron, J. S., Borland, D., Woosley, J. T., Xiaojun Guan, Schmitt, C., and Thomas, N. E. A method for normalizing histology slides for quantitative analysis. In *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 1107–1110, 2009.

Mansoor, A., Bagci, U., Foster, B., Xu, Z., Papadakis, G. Z., Folio, L. R., Udupa, J. K., and Mollura, D. J. Segmentation and image analysis of abnormal lungs at ct: Current approaches, challenges, and future trends. *RadioGraphics*, 35(4):1056–1076, 2015. PMID: 26172351.

Masters, D. and Luschi, C. Revisiting small batch training for deep neural networks. *CoRR*, abs/1804.07612, 2018.

McIntosh, A. The jackknife estimation method. 2016.

Messay, T., Hardie, R. C., and Tuinstra, T. R. Segmentation of pulmonary nodules in computed tomography using a regression neural network approach and its application to the lung image database consortium and image database resource initiative dataset. *Medical Image Analysis*, 22(1):48 – 62, 2015.

Milletari, F., Navab, N., and Ahmadi, S. V-net: Fully convolutional neural networks for volumetric medical image segmentation. *CoRR*, abs/1606.04797, 2016.

Mossel, J. How will ai empower the pathologists of the future? `https://ibex-ai.com/blog/how-will-ai-empower-the-pathologists-of-the-future-2/`, 2018.

Mukherjee, S., Huang, X., and Bhagalia, R. Lung nodule segmentation using deep learned prior based graph cut. 2017.

Márquez-Neila, P., Baumela, L., and Alvarez, L. A morphological approach to curvature-based evolution of curves and surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(1):2–17, 2014.

Nwankpa, C., Ijomah, W., Gachagan, A., and Marshall, S. Activation functions: Comparison of trends in practice and research for deep learning. *CoRR*, abs/1811.03378, 2018.

Otsu, N. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

Rakha, E. A., Aleskandarani, M., Toss, M. S., Green, A. R., Ball, G., Ellis, I. O., and Dalton, L. W. Breast cancer histologic grading using digital microscopy: concordance and outcome association. *Journal of Clinical Pathology*, 71(8):680–686, 2018.

Razzak, M., Naz, S., and Zaib, A. Deep learning for medical image processing: Overview, challenges and future. 2017.

Robbins, P., Pinder, S., de Klerk, N., Dawkins, H., Harvey, J., Sterrett, G., Ellis, I., and Elston, C. Histological grading of breast carcinomas: a study of interobserver agreement. *Hum. Pathol.*, 26(8):873–879, 1995.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

Ruder, S. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

Sakamoto, M., Nakano, H., Zhao, K., and Sekiyama, T. Lung nodule classification by the combination of fusion classifier and cascaded convolutional neural networks. pages 822–825, 2018.

Setio, A. A. A., Traverso, A., de Bel, T., Berens, M. S., van den Bogaard, C., Cerello, P., Chen, H., Dou, Q., Fantacci, M. E., Geurts, B., van der Gugten, R., Heng, P. A., Jansen, B., de Kaste, M. M., Kotov, V., Lin, J. Y.-H., Manders, J. T., Sóñora-Mengana, A., García-Naranjo, J. C., Papavasileiou, E., Prokop, M., Saletta, M., Schaefer-Prokop, C. M., Scholten, E. T., Scholten, L., Snoeren, M. M., Torres, E. L., Vandemeulebroucke, J., Walasek, N., Zuidhof, G. C., van Ginneken, B., and Jacobs, C. Validation, comparison, and combination of algorithms for automatic detection of pulmonary nodules in computed tomography images: The luna16 challenge. *Medical Image Analysis*, 42:1 – 13, 2017.

Shi, J. Lung nodule detection using convolutional neural networks. Master's thesis, EECS Department, University of California, Berkeley, 2018.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

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

Sullivan, J. Neural network from scratch: Perceptron linear classifier, 2017. [Online; accessed May 12, 2019].

Sutton, R. S. Two problems with backpropagation and other steepest-descent learning procedures for networks. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum, 1986.

Theodoridis, S. and Koutroumbad, K. Pattern recognition. pages 167–169, 2009.

Tompson, J., Goroshin, R., Jain, A., Lecun, Y., and Bregler, C. Efficient object localization using convolutional networks. pages 648–656, 2015.

Tretyakov, K. The mystery of early stopping, 2017. [Online; accessed May 12, 2019].

Vahadane, A., Peng, T., Albarqouni, S., Baust, M., Steiger, K., Schlitter, A., Sethi, A., Esposito, I., and Navab, N. Structure-preserved color normalization for histological images. pages 1012–1015, 2015.

van den Bergh, K. A., Essink-Bot, M. L., Bunge, E. M., Scholten, E. T., Prokop, M., van Iersel, C. A., van Klaveren, R. J., and de Koning, H. J. Impact of computed tomography screening for lung cancer on participants in a randomized controlled trial (NELSON trial). *Cancer*, 113(2):396–404, 2008.

van Dooijeweert, C., van Diest, P. J., Willems, S. M., Kuijpers, C. C. H. J., van der Wall, E., Overbeek, L. I. H., and Deckers, I. A. G. Significant inter- and intra-laboratory variation in grading of invasive breast cancer: A nationwide study of 33,043 patients in the Netherlands. *Int. J. Cancer*, 2019.

van Rikxoort, E., de Hoop, B., A Viergever, M., Prokop, M., and van Ginneken, B. Automatic lung segmentation from thoracic computed tomography scans using a hybrid approach with error detection. *Medical physics*, 36:2934–47, 2009.

Wahidi, M., A Govert, J., K Goudar, R., K Gould, M., and Mccrory, D. Evidence for the treatment of patients with pulmonary nodules: When is it lung cancer?: Accp evidence-based clinical practice guidelines (2nd edition). *Chest*, 132:94S–107S, 2007.

Wang, S., Zhou, M., Gevaert, O., Tang, Z., Dong, D., Liu, Z., and Tian, J. A multi-view deep convolutional neural networks for lung nodule segmentation. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1752–1755, 2017.

Wang, S., Zhou, M., Liu, Z., Liu, Z., Gu, D., Zang, Y., Dong, D., Gevaert, O., and Tian, J. Central focused convolutional neural networks: Developing a data-driven model for lung nodule segmentation. *Medical Image Analysis*, 40, 2017.

WHO. Cancer. `https://www.who.int/news-room/fact-sheets/detail/cancer`, 2018.

WHO. Cancer. `http://www.euro.who.int/en/health-topics/noncommunicable-diseases/cancer/policy`, 2018.

Wilbur, D. C., Madi, K., Colvin, R. B., Duncan, L. M., Faquin, W. C., Ferry, J. A., Frosch, M. P., Houser, S. L., Kradin, R. L., Lauwers, G. Y., Louis, D. N., Mark, E. J., Mino-Kenudson, M., Misdraji, J., Nielsen, G. P., Pitman, M. B., Rosenberg, A. E., Smith, R. N., Sohani, A. R., Stone, J. R., Tambouret, R. H., Wu, C.-L., Young, R. H., Zembowicz, A., and Klietmann, W. Whole-slide imaging digital pathology as a platform for teleconsultation: A pilot study using paired subspecialist correlations. *Archives of Pathology & Laboratory Medicine*, 133(12):1949–1953, 2009. PMID: 19961250.

Wingard, J. W. H., John R. and Jantz, M. A. How i manage pulmonary nodular lesions and nodular infiltrates in patients with hematologic malignancies or undergoing hematopoietic cell transplantation. 120:1791–1800, 2012.

Wu, J. and Qian, T. A survey of pulmonary nodule detection, segmentation and classification in computed tomography with deep learning techniques. *Journal of Medical Artificial Intelligence*, 2(0), 2019.

Yang, J., Veeraraghavan, H., Armato III, S. G., Farahani, K., Kirby, J. S., Kalpathy-Kramer, J., van Elmpt, W., Dekker, A., Han, X., Feng, X., Aljabar, P., Oliveira, B., van der Heyden, B., Zamdborg, L., Lam, D., Gooding, M., and Sharp, G. C. Autosegmentation for thoracic radiation treatment planning: A grand challenge at aapm 2017. *Medical Physics*, 45(10):4568–4581, 2018.

Zeiler, M. D. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

Zhou, Z., Siddiquee, M. M. R., Tajbakhsh, N., and Liang, J. Unet++: A nested u-net architecture for medical image segmentation. *CoRR*, abs/1807.10165, 2018.

Zhu, W., Vang, Y. S., Huang, Y., and Xie, X. Deepem: Deep 3d convnets with EM for weakly supervised pulmonary nodule detection. *CoRR*, abs/1805.05373, 2018.