Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

TU UB WIEN Universitätsbibliothek

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Convolutional Neural Networks for Bone Lesion Detection in Medical Imaging Data

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medizinische Informatik

eingereicht von

Matthias Perkonigg, BSc
Matrikelnummer 01129269

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ. Prof. Dipl.-Ing. Dr. Robert Sablating
Mitwirkung: Ass. Prof. Dipl.-Ing. Dr. Georg Langs

Wien, 23. April 2018

Matthias Perkonigg

Robert Sablating

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Convolutional Neural Networks for Bone Lesion Detection in Medical Imaging Data

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Medical Informatics

by

Matthias Perkonigg, BSc
Registration Number 01129269

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ. Prof. Dipl.-Ing. Dr. Robert Sablating
Assistance: Ass. Prof. Dipl.-Ing. Dr. Georg Langs

Vienna, 23rd April, 2018

Matthias Perkonigg   Robert Sablating

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

Erklärung zur Verfassung der Arbeit

Matthias Perkonigg, BSc
Pichlergasse 2, 1090 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 23. April 2018

Matthias Perkonigg

Danksagung

Am Ende dieser Arbeit, ist es an der Zeit ein Danke an jene zu richten die diese Arbeit unterstützt und in dieser Form ermöglicht haben.

Ich möchte mich an erster Stelle bei Georg Langs von der Medizinischen Universität Wien bedanken, der diese Arbeit betreut hat. Danke für die Anregungen und Ideen durch die diese Arbeit in dieser Form fertiggestellt werden konnte und für die Möglichkeit eine Arbeit in dem spannenden Bereich der medizinischen Bildverarbeitung zu schreiben. Ein weiterer Dank gilt Robert Sablating an der Technischen Universität Wien für die offizielle Betreuung und sein wertvolles Feedback während dem Verfassen der Arbeit.

Der größte Dank gebührt meinen Eltern, welche es mir durch ihre fortlaufende Unterstützung ermöglicht haben dieses Studium und diese Arbeit zu absolvieren. Ein Danke auch an meine Schwestern und Freunde welche während dieser Arbeit für Motivation und auch Ablenkung gesorgt haben.

Kurzfassung

Multiples Myelom ist eine Krebsart des blutbildenden Plasmas. Eines der Symptome das zur Diagnose und Einstufung von Patienten herangezogen wird sind Knochenläsionen. Diese Läsionen können mit Methoden der medizinischen Bildgebung sichtbar gemacht werden. Eine automatische Detektion der Knochenläsionen führt zu frühzeitiger Erkennung und Behandlung.

In dieser Arbeit wird eine Methode beschrieben, die durch multiples Myelom ausgelöste Läsionen in Computed Tomography und Magnetresonanz Bildern erkennt. Die Methode basiert auf Convolutional Neural Networks und das Konzept von Transfer Learning wird verwendet um Wissen, welches von natürlichen Bildern extrahiert wird, dazu zu verwenden um die Erkennung von Läsionen zu vereinfachen. Eine Methode um Ausschnitte mit drei Bildkanälen zu extrahieren wird vorgestellt. Diese Methode legt den Fokus jedes Kanals auf eine andere Information des medinzinischen Bilds. Ein Ansatz um Läsionen in Ganzkörperaufnahmen zu detektieren wird in dieser Arbeit vorgestellt.

Die Methoden werden auf 25 Computed Tomography und 25 Magnetresonanz-Volumen, in denen Läsionen annotiert sind, trainiert und evaluiert. Die Arbeit beinhaltet außerdem eine qualitative Analyse der Detektionsmethode.

Die Resultate zeigen, dass die Erkennung von Läsionen in neuen Ganzkörperbildern möglich ist. Allerdings zeigt sich, dass die Anzahl an Fehldetektionen hoch ist. Die Gründe dafür werden im Zuge der Arbeit diskutiert. Transfer Learning und auch der Ansatz der Extrahierung von Bildausschnitten in drei Kanäle verbessern die Ergebnisse der Methode.

Abstract

Multiple Myeloma is a type of cancer in plasma cells. One of the symptoms used for diagnosing and staging Multiple Myeloma patients are bone lesions that are visible in medical imaging scans. Automatic detection of bone lesions leads to earlier detection and treatment.

In this thesis an approach to detect bone lesions caused by Multiple Myeloma in computed tomography and magnetic resonance imaging is described. The method uses a convolutional neural network to detect lesions. Transfer learning is used to transfer knowledge from a natural image task to the task of detecting bone lesions in medical images. A three channel patch extraction method, in which each input channel of the network focuses on different parts of the information encoded in the medical image, is introduced. A sliding window approach is used in the thesis to detect lesions in whole body scans.

The method is trained and evaluated on a set of 25 computed tomography and 25 magnetic resonance volumes with annotated bone lesions. A qualitative analysis of the volume parsing approach is carried out.

Results show that it is possible to detect lesions in new whole body scans. However, numerous false positives are also detected and reasons for this misclassifications are analyzed. Transfer learning improves the results of the thesis and also the three channel patch extraction is a valuable addition to the method.

Contents

Kurzfassung **ix**

Abstract **xi**

Contents **xiii**

1 Introduction **1**
1.1 Motivation 1
1.2 Aim of the Thesis 2
1.3 Outline of the Thesis 2

2 State of the Art: Machine Learning Basics and Neural Networks **5**
2.1 Basic Notions of Machine Learning 5
2.2 Basic Definitions of Neural Networks 8
2.3 Training of a neural network 11
2.4 Summary 19

3 State of the Art: Convolutional Neural Networks **21**
3.1 Convolution 21
3.2 Key ideas of a CNN 22
3.3 Regularization Strategies 27
3.4 Design of a Convolutional Neural Network 28
3.5 Convolutional Neural Networks in Medical Imaging 31
3.6 Summary 32

4 State of the Art: Transfer Learning **33**
4.1 Basic Concept of Transfer Learning 33
4.2 Transfer Learning for Convolutional Neural Networks 37
4.3 Transfer Learning for Convolutional Neural Networks in Medical Imaging 40
4.4 Summary 41

5 Preliminary Experiments **43**
5.1 Dataset for the preliminary experiments 43
5.2 Experiment 1: CNN architecture 43

5.3 Experiment 2: Transfer learning strategy 44
5.4 Summary 45

6 Methodology **47**
6.1 Bone masking 47
6.2 Convolutional Neural Network - Architecture 47
6.3 Patch Extraction Strategies 49
6.4 Learning Protocols 52
6.5 Model training 53
6.6 Classifying new imaging data 54
6.7 Volume parsing 54
6.8 Summary 55

7 Evaluation **57**
7.1 Data 57
7.2 Evaluation of classification accuracy on image patches 59
7.3 Evaluation of volume parsing 61
7.4 Experiment Setup and Parameter Settings 62
7.5 Summary 64

8 Results on the VISCERAL Dataset **67**
8.1 Results on computed tomography data 67
8.2 Results on MRI data 77
8.3 Summary 87

9 Summary and Discussion **89**
9.1 Ability to detect lesions 89
9.2 Transfer Learning vs. Random Initialization 89
9.3 Single Channel Image Patches vs. Three Channel Image Patches . . . 90
9.4 Limitations 91
9.5 Summary 91

10 Conclusion and Future Work **93**
10.1 Future Work 94

Bibliography **95**

CHAPTER 1

Introduction

In this first chapter, an introduction to the disease of multiple myeloma is given and the motivation to detect bone lesions is described. Subsequently, the aim of the thesis is discussed. Finally, an outline of the structure of the thesis is given.

1.1 Motivation

Multiple Myeloma is a type of cancer with the proliferation of neoplasma cells in the bone marrow [11]. It accounts for 1.3% of all cancer diagnoses and about 15% of all new hematologic malignacies [48]. One of the criteria for symptomatic myeloma, as agreed by the International Myeloma Working Group in 2003 is evidence of end-organ damage. This damage is manifested by increased calcium (C), renal insufficiency (R), anamemia (A), and bone lesions (B), or *CRAB* [1].

Bone lesions are the most common symptom for multiple myeloma, 70%-90% of multiple myeloma patients develop osteolytic lesions [11] [18]. The destruction of the bone is a major cause for morbidity and mortality, since further progression is not always affected by chemotherapy [18]. Through the detection of bone lesions multiple myeloma patients can receive treatment earlier and thus have a higher chance of survival [48]. Also skeletal complications caused by multiple myeloma can be found by the detection of bone lesions [18].

Different imaging modalities are used to diagnose multiple myeloma. Radiographs were used as the initial imaging evaluation, however the sensitivity to detect bone lesions is low. More advanced techniques like Computed Tomography (CT), Magnetic Resonance Imaging (MRI) or Positron Emission Tomography (PET) are used to detect bone lesions with higher accuracy [11] [21].

In this thesis, bone lesions in CT and MRI data are automatically detected. CT has the ability to detect small osteolytic lesions faster than standard radiography [18]. It also allows a 3D reconstruction of images and helps to plan radiotherapy and surgical

intervention [18].
MRI has a higher sensitivity than CT and radiography and it can discriminate between normal marrow and myeloma [21]. Both modalities are used in clinical practice to detect bone lesions [18].

Computer-Aided Detection (CAD) systems have been developed to help radiologists during their diagnosis and staging of cancer patients with various types of cancer [22]. However, the reliable automatic detection of bone lesions caused by multiple myeloma in CT and MRI data is, to the knowledge of the author, not possible yet.

1.2 Aim of the Thesis

This thesis aims to develop a method to detect bone lesions in Multiple Myeloma patients in two modalities: CT and T2 weighted MRI.
For this purpose a Convolutional Neural Network (CNN) approach is used. In this thesis two learning protocols are evaluated: learning from scratch and transfer learning. Transfer learning is a method that aims to transfer knowledge learned in a source task to improve learning in a target task [69]. To train the CNNs a patch extraction method is developed which extracts 2D image patches from 3D CT and MRI volumes. After the networks are trained, a sliding window approach is implemented to detect lesions in whole body volumes.

Two main questions are answered in the course of this thesis:

- Is it possible to detect bone lesions in CT and MRI data with the help of convolutional networks?
- Does transfer learning help to improve the performance of a detection with CNNs, even though the network is pre-trained on non medical natural images?

In order to answer these questions different networks are trained to compare results of random initialization to transferring network parameters from a source task. Due to the different nature of the CT and MRI data the developed methods should be evaluated on CT and MRI data separately.

Generative Adversarial Networks (GAN) are an alternative to using plain convolutional networks, until recently there were no applications using GANs in medical imaging [38]. However, they are adapted for use in medical imaging recently (e.g. [57], [73]). This thesis focus on transfer learning approaches, the application of GANs to the detection of bone lesions is out of the scope of this thesis.

1.3 Outline of the Thesis

In the following, an outline of the thesis is given.

- **Chapter 2** gives an introduction to basic concepts of machine learning which are necessary to comprehend the following chapters of the thesis. Also, Neural Networks as the base on which CNNs are built upon are introduced.
- **Chapter 3** explains CNNs in more detail. The key concepts of convolutional networks are explained and hyperparamters are introduced. Also, an overview of the state-of-the-art of CNNs used in the domain of medical imaging is given.
- **Chapter 4** introduces the concept of transfer learning in general and specifically for convolutional networks. An overview of the state-of-the-art of using transfer learning in medical imaging is given.
- **Chapter 5** documents two preliminary experiments, which are carried out to decide on a final method that is used in this thesis.
- **Chapter 6** discusses the methodology used in the implementation of the thesis. Patch extraction, learning protocols and the volume parsing approach used are explained in detail.
- **Chapter 7** gives details about how the outcome of the thesis is interpreted and evaluated.
- **Chapter 8** gives results for all evaluations run on CT and MR data. Quantitative results on image patches are given and the volume parsing is evaluated qualitatively.
- **Chapter 9** summarizes the thesis and discusses the outcome of the work. Also, the research questions stated in Section 1.2 are answered.
- **Chapter 10** concludes the thesis and gives an insight into future work.

CHAPTER 2

State of the Art: Machine Learning Basics and Neural Networks

In this chapter, basic definitions and notions of machine learning are explained. Afterwards, neural networks on which Convolutional Neural Networks are built upon are described. The basic architecture of such networks as well as the training process are explained.

2.1 Basic Notions of Machine Learning

In this section, basic notions of machine learning are explained. These notions are used in the following chapters to understand the principle of the methods and outcomes of the thesis.

2.1.1 Machine Learning

In general, machine learning algorithms, and as a part of them deep learning algorithms such as CNNs try to learn from labeled data. The aim is to learn a mapping from input data to an output value [8].
Mitchell [44] defines machine learning with a concept of experience E from which a computer learns in respect to some task T and a performance measure P. An algorithm is said to learn from E if the performance on T measured with P improves with the experience E [44]. The experience of a machine learning algorithm is the *training set* $\{x_1, \cdots x_n\}$ given to the algorithm during the training phase.

There are a variety of tasks that a machine learning algorithm can be used for, e.g. classification (see Section 2.1.3) [27]. This task is solved by learning a mapping function

$y = f(x)$. The performance measure in such a task is commonly the proportion of samples for which the algorithm produces the correct output. This performance measure is called *accuracy* and is measured on a *test set* which is independent of the training set [8].
Based on the experience available to a machine learning algorithm, it learns in a supervised or unsupervised environment. For a discussion of other learning protocols like reinforcement learning and semi-supervised learning see [27].

2.1.2 Supervised learning vs. Unsupervised learning

Machine learning models can be trained in a supervised manner. That means that the training data used consists of input vectors along with corresponding targets or labels [8]. The goal is to learn how to predict some target vector $\mathbf{y}$ from the input vector $\mathbf{x}$. During training a teacher 'supervises' the machine learning system and provides target vectors $\mathbf{y}$ to the learner [59]. With supervised learning the aim is to learn a probability distribution $p(\mathbf{y}|\mathbf{x})$, which is the probability to output value $\mathbf{y}$ given the input $\mathbf{x}$.
In contrast, unsupervised learning means that the algorithm is trained on input vectors without any corresponding targets [8]. The goal is to discover useful properties in the structure of the given training data. Examples for unsupervised learning approaches are clustering or density estimation [27]. Unsupervised methods are out of the scope of this thesis.
This thesis deals with a supervised classification problem that means that the goal is to predict the right label for each input vector.

2.1.3 Classification

Classification aims to assigning to which of m categories or classes an input belongs [59]. During training it learns a function $f : \mathbb{R}^n \rightarrow \{1, \ldots, m\}$ which can assign a category to a given input $\mathbf{x}$: $\mathbf{y} = f(\mathbf{x})$ [27].
Classification algorithms sometimes also assign a probability to their output. For a binary classification (e.g. positive/negative) it is sufficient that f produces a single output value which represents the probability of the input being a positive.

2.1.4 Hyperparameters

Hyperparameters are parameters of a machine learning algorithm that have to be set before training. Those parameters are used to alter the algorithm's behaviour [27]. The hyperparameters are not learned nor altered during training. Most hyperparameters govern the capacity of a model [27]. A hyperparameter has to be set prior to applying data to an alogrithm, since it is a parameter that can not be chosen by the algorithm itself [7]. The tuning of the hyperparameters should be done on a separated validation set that is neither used during training nor to assess the generalization error of the learned model [27]. Hyperparameters can be continuous, like the learning rate of a neural network or discrete like the number of hidden layers of a neural network.

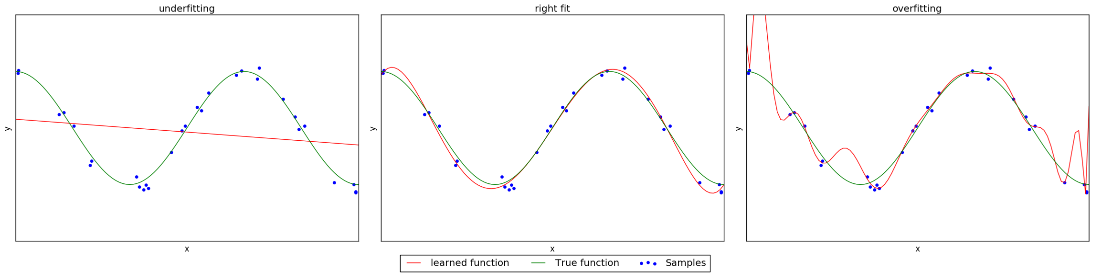


Figure 2.1: Over- and underfitting: A polynomial function should be fitted to a sample of points. Those samples are generated by evaluating the true function at randomly chosen x and adding some noise to the result y. *Left:* A linear function cannot approximate the true function, it underfits the task. *Center:* A polynomial of degree 5 fits well to the true function. *Right:* A polynomial of degree 15 overfits the samples.

There are different strategies how to train hyperparameters. Some are set by common understanding of the model used and are tested manually by the developer of the machine learning system [27]. Another posible way to set parameters are search strategies like grid search or random search which are used to find suitable parameter values [27]. With grid search the developer chooses a finite set of values for each hyperparameter. The algorithm then tries out each joint hyperparameter setting on a validation set to determine the best joint setting. The setting performing best in this search is chosen for training the model on the training set [27].
With random search a marginal distribution is defined for each hyperparameter. The algorithm runs for a given number of iterations. In each iteration a value is drawn for each hyperparameter and the setting is tested. The setting with the best performance is chosen for training the model [27].

2.1.5 Overfitting and underfitting

Overfitting and underfitting occurs when a model does not have the right capacity to solve a task. Consider Figure 2.1 for an example. If the capacity of the function is too low it underfits the problem. If it is too high the learned function fits perfectly to the training samples, but its generalization is poor [8].

During training of a machine learning model two goals should be reached. First, to decrease the training error and secondly to keep the gap between training and test error as small as possible. If a machine learning algorithm is not able to decrease the training error, the model *underfits* the task [8]. If the gap between training and test error gets too large, *overfitting* occurs [27].
The capacity of a model is usually controlled through the choice of its hyperparamters. In the example in Figure 2.1 the degree of the polynominal function is a hyperparameter which controls the capacity [8].

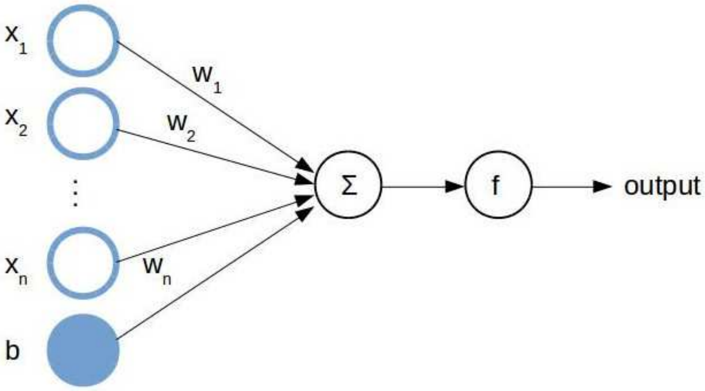


Figure 2.2: A simple neuron: $\{x_1, \dots, x_n\}$ are the inputs to the neuron. b is the bias and $\{w_1, \dots, w_n\}$ is a set of individual weights for each input. The output is computed according to Equation 2.1.

2.2 Basic Definitions of Neural Networks

Neural networks are built on artifical neurons [41] on which *Rosenblatt* developed his concept of the perceptron an early variant of a neural network [54]. Neural networks group neurons in a graph-like structure (see Figure 2.5) to solve the problem of approximating some function f that maps from an input to an output $y = f(x)$.
In this section, basic notions and definitions for neural networks are given. First, artificial neurons are described, building upon those neurons feedforward networks are discussed.

2.2.1 Artificial neurons

Artificial neurons were introduced by McCulloch and Pitts in 1943 [41]. Based on this concept the neurons used in neural networks were adapted [54]. Neurons are the basic element of neural networks. The structure of a neuron in shown in Figure 2.2. An input vector $\mathbf{x} = \{x_1, \dots, x_n\}$ is given to the neuron, the data is processed and an output is computed. The output of a neuron is computed from multiple inputs x_i as follows:

$$output = f(\sum_{i=1}^{n} w_i x_i + b) \tag{2.1}$$

Where w_i are the weights, b is the bias and $f(.)$ a non-linear activation function. Each input x_i is weighted by an individual weight w_i. The added bias can be interpreted as an offset in the data [8].

Weights and bias are combined linearly. This linear combination $(w_i x_i + b)$ is called the *activation* of the neuron. Since the aim of neural networks is not only to learn linear

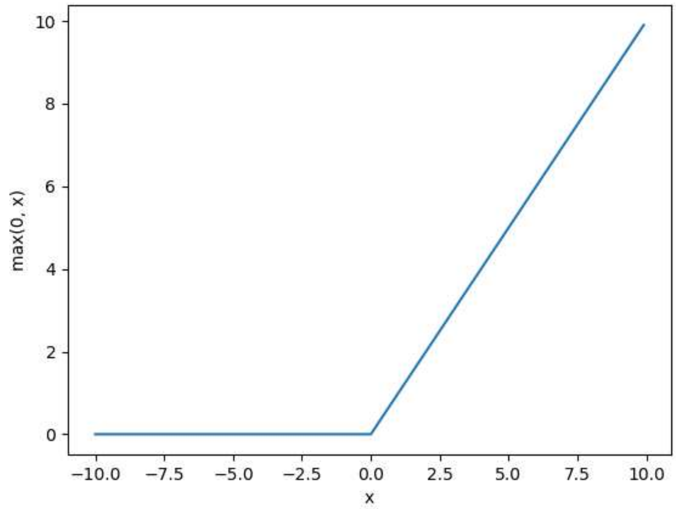


Figure 2.3: Rectified linear unit (ReLu)

combinations, but rather to learn non-linear mappings from input to output, a non-linear activation function is needed. Typical choices of such activation functions are described in the following.

2.2.2 Activation function

A common choice for the activation function in modern networks is the rectified linear unit (ReLu) [27] defined as:

$$f(x) = max(0, x) \tag{2.2}$$

The output of the ReLu function is depicted in Figure 2.3. The use of ReLu has the benefit that the function is similar to linear units and thus easy to optimize, because the derivatives remain high whenever the unit is active. Because the gradient is consistent, the gradient direction of a ReLu is more useful than with other activation functions [27]. An explanation why the gradient of an activation function is important is given in Section 2.3.2.

Other popular activation functions include the logistic sigmoid function

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

or the hyperbolic tangent function

$$f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.4}$$

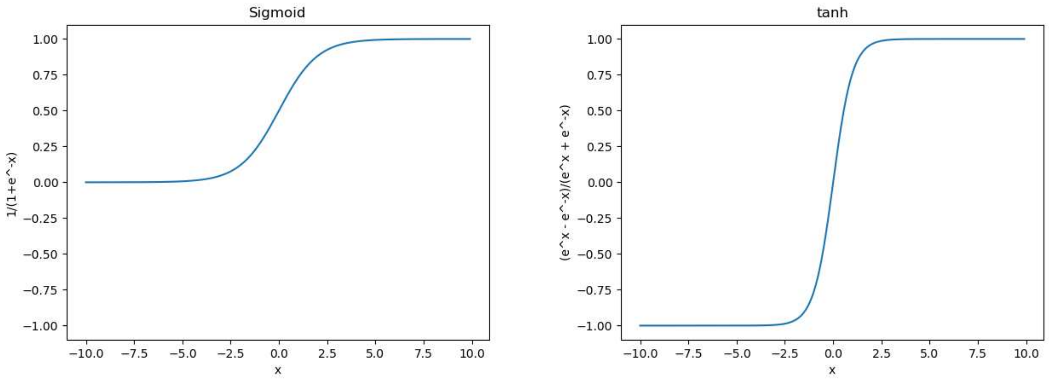


Figure 2.4: *Left:* Logistic sigmoid. *Right:* hyperbolic tangent function (tanh)

The sigmoid function is commonly used as activation function of output units for predicting binary variables. These functions are shown in Figure 2.4.

2.2.3 Feedforward Neural Networks

Artificial neurons as described in the previous sections are the essential building blocks for feedforward neural networks, which are discussed in this section.

Feedforward Neural Networks or Multi-Layer Percptrons (MLP) are used to approximate some unknown function f^*. When used as a classifier, the function $y = f^*(x)$ is a mapping from an input x to an output category y. This approximation is learned by defining a mapping $y = f(x; \theta)$ and then trying to learn the parameters θ that yield the best results [27].

A neural network can be interpreted as a directed graph where the nodes correspond to neurons and the edges represent links between them [59]. Such a graph is shown in Figure 2.5.

The neurons are grouped into layers. The first layer is called the input layer, the last layer the output layer. All layers inbetween are called hidden layers, because their desired activiations are not in the training data [27]. A feedforward neural network is fully connected, which means that each neuron of a layer is connected to each neuron in the following layer. Also, no feedback connections, which connect the outputs of neurons to itself, are used in feedforward neural networks.

To understand how the output of a simple neural network is calculated consider Figure 2.5. The model consists of one input layer L_1, one hidden layer L_2 and one output layer L_3. The input is fed to the input layer and the output of the hidden layer are calculated as follows:

$$a_1 = f(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + b_1^{(1)}) \tag{2.5}$$

$$a_2 = f(W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + b_2^{(1)}) \tag{2.6}$$

$$a_3 = f(W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + b_3^{(1)}) \tag{2.7}$$

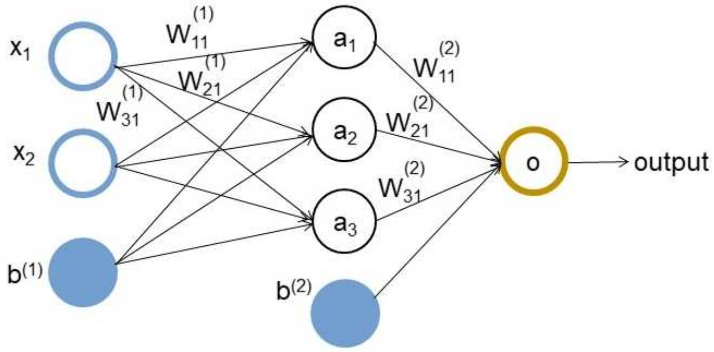


Figure 2.5: A simple neural network consisting of one input layer, one hidden layer and one output layer.

where f is the activiation function of the neurons in the hidden layer [8].
The output of the network is usually computed with a different activation function σ than those of the hidden layers.
The output is calculated as:

$$o = \sigma(W_{11}^{(2)} a_1 + W_{12}^{(2)} a_2 + W_{13}^{(2)} a_3 + b_1^{(2)}) \tag{2.8}$$

The value of the output corresponds to the result of the approximation of the function f^*. Suitable parameters (weights and biases) can be found by training the network in a training phase. This training is described in the following section.

2.3 Training of a neural network

Training a neural network describes the process of finding suitable weight parameters W and biases b to optimize the approximation of the unknown function f^*.
This training is done by

- Finding a cost function,
- optimizing the weights to minimize the cost with gradient descent and
- the use of backpropagation to efficiently compute the gradients.

Those three main steps are described in the following.

2.3.1 Cost function

To train a neural network a cost function is needed. During training the aim is to find weights that minimize this cost function. The cost function is usually the cross-entropy between the training data distribution and the predictions of the model. The cross-entropy is derived from the maximum likelihood estimation, which is commonly used in neural networks [27].

Maximum Likelihood Estimation

Consider n examples $X = \{x_1, x_2, ..., x_n\}$ drawn from the true, but unknown probability distribution $p_{data}(x)$. $p_{model}(X;\theta)$ is a parametric set of probability distributions indexed by θ [27]. For neural networks this θ are the weights and biases of the model.
The maximum likelihood for θ is then given by:

$$\theta_{ML} = \underset{\theta}{argmax} p_{model}(X;\theta) \tag{2.9}$$

$$= \underset{\theta}{argmax} \prod_{i=1}^{m} p_{model}(x_i;\theta) \tag{2.10}$$

This can be reformulated as:

$$\theta_{ML} = \underset{\theta}{argmax} \mathbb{E}_{x\sim\hat{p}_{data}}[\log p_{model}(x;\theta)] \tag{2.11}$$

which is an expectation with respect to the empirical distribution $\hat{p}_{data}$ defined by the training data [27].

Cross-Entropy

The cross-entropy, or negative log-likelihood measures the similarity between two probability distributions $P(x)$ and $Q(x)$ and is defined as:

$$H(P,Q) = -\mathbb{E}_{x\sim P} \log Q(x) \tag{2.12}$$

Equation 2.12 can be reformulated to measure the similarity of the output of a neural network and the distribution of the data $p_{data}(x)$. The output of a neural network can be interpreted as defining a distribution $p(y|x;\theta)$, in which y is the output, x the input and θ the parameters, namely weights and biases. The negative log-likelihood or cross-entropy between training data and the predictions of a parametric model is then given by:

$$J(\theta) = -\mathbb{E}_{x,y\sim\hat{p}_{data}}[\log p_{model}(y|x)] \tag{2.13}$$

In Equation 2.13 it can be seen that the cost depends on the model used. Here, two cross-entropy functions are introduced, which are used with neural networks.
If the network should only predict a single value $P(y = 1|x)$ *binary cross-entropy* is used. For the prediction of a single probability value a sigmoid output unit is used. The output of the unit is defined by:

$$\hat{y} = \sigma(w^\top h + b) \tag{2.14}$$

with σ being the sigmoid function defined in Equation 2.3. The output of σ is the probability $P(y = 1|x)$. The cost function for a binary classification problem can then be written as:

$$\begin{aligned} J(\theta) &= -logP(y = 1|x) \qquad (2.15) \\ &= -(y\log(\hat{y}) + (1 - y)\log(1 - \hat{y})) \qquad (2.16) \end{aligned}$$

where y is the true label (0 or 1) and $\hat{y}$ is the probability value the network computed according to Equation 2.14.
If the network is build to do a multiclass classification by predicting a class label of n possible values *categorical cross-entropy* is used as cost function [27]. To produce a valid probability distribution over an output vector $\mathbf{y}$, where $y_i = P(y = i|x)$ a softmax function is used to normalize the output, the softmax function is given by,

$$softmax(z)_i = \sigma_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e_j^z} \tag{2.17}$$

where z_i is the activation of the i-th output unit [8].
The softmax function normalize the output of the units so that a valid probability distribution fulfilling the constraints $0 \leq \sigma_i \leq 1$ and $\sum_{j=1}^{n} \sigma)_i = 1$ is generated. Following the negative log-likelihood as before, a cost function can be written as:

$$J(\theta) = \sum_{i=1}^{n} \sum_{j=1}^{m} y_{ij} \sigma_j \tag{2.18}$$

where y_{ij} is a one hot encoded target vector defined as:

$$y_{ij} \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases} \tag{2.19}$$

2.3.2 Gradient Descent

With the cost functions defined in the previous section, an algorithm to train the model by changing the parameters and thus minimizing the cost is needed.
This is commonly achieved by using some form of gradient descent. The basic idea of gradient descent is to use the first derivative of a convex function to determine how to change the parameters to minimize the function [59]. Consider a function $y = f(x)$, then the first derivate of this function $f'(x)$ gives the slope of f at point x [27]. For a function with an input vector instead of a single parameter the gradient of the function $\nabla_\theta f(\theta)$

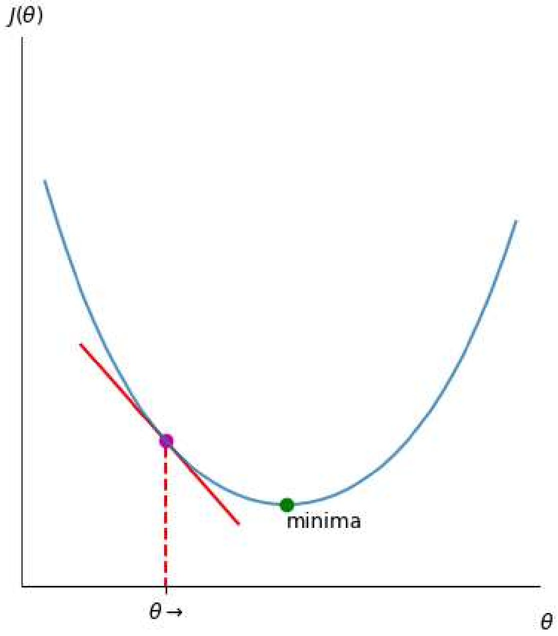


Figure 2.6: Gradient descent - The cost function $J(\theta)$ is evaluated at a position θ, the gradient is calculated and a step in the negative direction of the gradient is taken. In this case the step is taken to the right in direction of the minima. After the step the same procedure is iteratively repeated until a critical point is reached.

where θ is a vector. The gradient is defined as the vector of the partial derivatives of the function f in respect to the elements of the parameter vector.
Since the error functions discussed in the previous section are smooth, continous and convex functions we can use the gradient to determine how to change the weights of our neural network in order to minimize the cost function [8].
This is called gradient descent [8]. The idea of gradient descent is shown in Figure 2.6
The idea is to calculate the gradient of the cost function and make a small step in the negative direction of this gradient to calculate a new point θ', which hopefully brings the parameter settings closer to a minimum:

$$\theta' = \theta - \epsilon \nabla J(\theta) \tag{2.20}$$

This process is repeated iteratively to reach a minimum of the function. It can not be guaranteed that gradient descent arrives at a minimum. The algorithm determines when a critical point is reached.

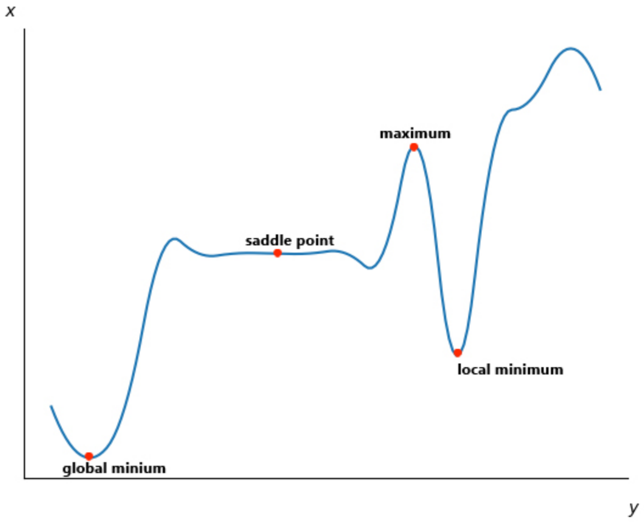


Figure 2.7: Critical points of a function - Gradient descent finds one of these critical points. It can not be guaranteed that the global minimum is found.

Critical Points

Points where the gradient of the function evaluates to zero are called critical points. When gradient descent arrives at a critical point, no decision can be made in what direction the parameters θ should be changed and thus the algorithm ends. Critical points can be minimum, maximum and saddle points [8]. See Figure 2.7 for examples of critical points. A local minimum is a point which is lower than all neighbouring points. A minimum is called global if the point evaluates to the absolute lowest value of the function. Similarly, a maximum is a point which is higher than all neighbouring points. A saddle point is a point where the gradient is zero, but the point is neither a maximum nor a minimum. The goal of gradient descent is to find a global minimum. There is no guarantee that a global minimum is found. In deep learning practice, often a point that is low but not necessarily minimal is found, which is sufficient in most cases [27].

Stochastic Gradient Descent (SGD)

A problem in using gradient descent directly in training a neural network is that it is computationally expensive to calculate the cost function over the whole training set. One solution to this is the *Stochastic Gradient Descent (SGD)*. The idea is to compute the gradient after evaluating the cost function on a single sample of the training set. This can be done because the cost function used while training a neural network can be decomposed as a sum of a per-sample loss function. Following Section 2.3.1 the negative log-likelihood is used, which can be written as:

$$J(\theta) = \mathbb{E}_{x,y \sim \hat{p}_{data}} L(x, y, \theta) = \frac{1}{n} \sum_{i=1}^{n} L(x_i, y_i, \theta) = \frac{1}{n} \sum_{i=1}^{n} L_i \tag{2.21}$$

where $L(x, y, \theta) = -\log p(y|x; \theta)$. We can then calculate the gradient for those per-sample loss function separatedly [27]. Stochastic gradient descent makes an update to the parameter vector based on one training sample at a time resulting in:

$$\theta^{\tau+1} = \theta^{\tau} - \eta \nabla L_i(\theta^{\tau}) \tag{2.22}$$

where η is called the learning rate and τ denotes the iteration [8]. L_i is the loss of training sample i.

SGD with minibatches

To update the parameter setting after each sample is computationally expensive and the cost function of a single sample is often not enough to capture the cost function of the training data [8].
In practice an update to the parameter vector is usually done after evaluating so called *minibatches*. At every iteration of the algorithm we sample a set $\mathbb{B} = \{x_1, x_2, ..., x_m\}$ from the training set, with m being a small minibatch size. Then the estimate for the gradient is given by

$$g = \frac{1}{m} \nabla \sum_{i=1}^{m} L(x_i, y_i, \theta) \tag{2.23}$$

The averaged gradient g is then used in Equation 2.22 to determine how to change the parameters [27].

SGD with minibatches and momentum

Furthermore, *Polyak* introduced a method to accelarate learning to be used with SGD called *momentum* [49]. With momentum the speed and direction at which parameters move trough parameter space are represented by a variable called velocity v. The idea is that if the gradient direction remains steady over time a larger step is taken in this direction. This is done by setting the parameter v to an exponentially decaying average of the negative gradient:

$$v = \alpha v - \eta \frac{1}{m} \nabla \sum_{i=1}^{m} L(x_i, y_i, \theta) \tag{2.24}$$

where α is a hyperparameter controlling how quickly the the contribution of previous gradient exponentially decay [27]. The velocity is then used to update the parameters

$$\theta^{\tau+1} = \theta^{\tau} + v \tag{2.25}$$

The final minibatch SGD algorithm with a fixed number of epochs e with momentum is given in Algorithm 2.1.

Algorithm 2.1: Stochastic Gradient Descent (SGD) with minibatches and momentum

```
Input: $\eta$, learning rate
Input: $\alpha$, momentum parameter
Input: $\theta$, initial parameters
Input: $v$, inital velocity
Input: $e$, number of training epochs
Input: $m$, size of minibatch
Output: $\theta$, final parameters
for $i \leftarrow 1$ to $e$ do
    Sample minibatch $\{x_i, \dots, x_m\}$ with corrsponding labels $j_i$
    Compute gradient: $g = \frac{1}{m}\nabla \sum_{i=1}^{m} L(x_i, y_i, \theta)$
    Compute velocity: $v = \alpha v - \eta\frac{1}{m}\nabla \sum_{i=1}^{m} L(x_i, y_i, \theta)$
    Apply update: $\theta = \theta + v$
end
```

This training strategy is also used in the method developed in this thesis. Modern neural networks use some variant of SGD, a detailed discussion can be found in [27].

2.3.3 Backpropagation

Evaluating the gradient for a loss function numerically is computationally expensive. Therefore, an algorithm is needed to efficiently compute those gradients in a neural network. This is done by backpropagation of the errors made by the network [56].

During training the information of an input x is propagated forward through the network to compute a loss $J(\theta)$ at the output layer. The backpropagation algorithm lets the computed information about the loss flow backwards through the network to compute the gradient [27]. In this section, the backpropagation algorithm as used in neural networks to compute the gradient of the loss function is explained. Backpropagation is not restricted to that use, a more extensive explanation and discussion of backpropagation in general can be found in [27].

First, a forward pass to compute the supervised loss is executed. Algorithm 2.2 shows this forward pass. The algorithm shows how a single input vector $\mathbf{x}$ flows through the network and how the cost J of this input is calculated [27].

Starting from the result J of this algorithm a backward pass can be executed to calculate the gradient of the cost function at a position x. Algorithm 2.3 shows the backpropagation where the information about the loss flows backwards through the neural network to compute the gradients [8].

Algorithm 2.2: Forward-pass of the backpropagation algorithm, adapted from [27]

```
Input: $l$, the depth of the network
Input: $\theta_i, i \in 1, ...., l$, the parameters of the model
Input: $\mathbf{x}$, the input to process
Input: $\mathbf{y}$, the target output
Output: J, the loss of the forward pass
$h_0 = \mathbf{x}$
for $k \leftarrow 1$ to $l$ do
    $a_k = b_k + W_k h_{k-1}$
    $h_k = f(a_k)$
end
$\hat{y} = h_l$
$J = L(\hat{y}, y)$
```

Algorithm 2.3: Backward-pass of the backpropagation algorithm, adapted from [27], [8]

```
Input: Inputs are all variables defined in 2.2
Output: Gradients of the activations $a_k$ of all layers. Gradients of all biases and
        weights in the network.
Setting the error terms for the output unit:
$\mathbf{g} = \nabla_{\hat{y}} J = \nabla_{\hat{y}} L(\hat{y}, y)$
for $k \leftarrow l - 1, l - 2$ to $1$ do
    $\mathbf{g} = \nabla_{a_k} J = \delta^{k+1} \odot f'(a_k)$
    Compute gradients on bias and weights, those can be used for gradient descent
      updates
    $\nabla b_k J = \mathbf{g}$
    $\nabla w_k J = \mathbf{g} h_{(k-1)}^{\top}$
    Propagate gradients to next lower-level:
    $\mathbf{g} = W_k^{\top} \mathbf{g}$
end
```

The information about the cost flows backwards layer-by-layer until it reaches the input layer. The gradients computed in line 6 and 7 can be used to apply an update to the corresponding bias and weight terms of the current layer k during stochastic gradient descent.

2.4 Summary

In this section basic notions of machine learning and neural networks are described. Different learning paradigms as well as a specific task, classification that is relevant to this thesis are introduced. The importance of hyperparameters and the problem of over- and underfitting is described.
For neural networks activations functions and the architecture of feedforward neural networks are discussed. Important for the thesis is the description of the training process, because this knowledge is needed to understand how convolutional networks are trained. Three steps are important for training a neural network: finding a cost function, using gradient descent and the backpropagation algorithm, which enables the fast computation of gradients of the cost function. Two specific cost functions which are relevant to the thesis are introduced the binary cross-entropy and the categorical cross-entropy. For gradient descent the practical algorithm of stochastic gradient descent with minibatches and momentum as used for training in the course of this thesis is described.

CHAPTER 3

State of the Art: Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a special type of neural network with at least one convolutional layer. They are especially suitable for processing data with a grid-like structure [27]. Different domains can be processed by CNNs e.g. language sequences as 1D arrays, images and audio signals as 2D and videos or medical imaging data as 3D grids [35]. The first successful application of CNNs to real world application was a zip code recognition by *LeCun et al.* in 1990 [37]. *LeCun et al.* introduced their famous LeNet-5 in 1998, a CNN with 7 levels that were used for handwritten digit recognition [36].
CNNs gained renewed research interest after *Krizhevsky et al.* won the ImageNet object recognition challenge in 2012 by using a convolutional network [34]. Since then CNNs became state-of-the-art in many computer vision tasks such as classification, object detection or segmentation [35].

In this chapter, the basics of CNNs are explained. First, the mathematical concept of a convolution is described. Next, the main ideas of CNNs, sparse connectivity, parameter sharing and pooling are discussed. Afterwards, an introduction to regularization strategies for CNNs related to this thesis is given. Then, architectures and hyperparameters commonly used in context of CNNs are introduced and finally, an overview of the state of the art of CNNs used for medical imaging is given.

3.1 Convolution

In general, a convolution is the integral of two functions f and g, with one of the functions reversed and shifted. A convolution is written as $f * g$ and defined as:

$$s(t) = (f * g)(t) = \int f(\tau)g(t-\tau)d\tau \tag{3.1}$$

Since data in a computer is sampled and discretized we can use a discrete form of the convolution equation:

$$s(t) = (f * g)(t) = \sum_{\tau=-\infty}^{\infty} f(\tau) g(t - \tau) \tag{3.2}$$

With the terminology used with CNNs the first argument (in Equation 3.2 the function f) is the input. The second argument is denoted as kernel and is basically a weighting function. The output of the convolution is called feature map [27].
In the following, the explanations use an 2D image as input for simplicity and because this thesis deals with 2D images. The equations can be extended to any other grid-like structure [27].

Assuming a 2D input image I and a 2D kernel K, we can do the convolution over more than one axis at a time resulting in the following equation:

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n) K(i - m, j - n) \tag{3.3}$$

Often in convolutional network implementations cross-correlation, a related function without flipping the kernel is used instead of a convolution. However, the operation is still called convolution in practice [27]. Cross-correlation is defined as:

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n) K(i + m, j + n) \tag{3.4}$$

3.2 Key ideas of a CNN

CNNs are designed to process grid-like data and to take advantage of the properties of natural signals [35]. They are inspired by neuroscientific experiments carried out by *Hubel and Wiesel*, who worked with cats to figure out how the brain responds to images [30]. They found that neurons of the early visual system respond to simple patterns like a line at a particular place on the retina. Along the visual pathway those patterns get increasingly more complex to construct an image [30]. Like this visual pathway, CNNs detect local features in early layers while more complex structures are detected in deeper layers of the network. A more in-depth discussion about the neuroscientific basics for convolutional networks can be found in [27]. From the outcomes of the cat experiment the key ideas of CNNs were developed.

First, in natural signals values in a local proximity to each other are often highly correlated, those local features can be detected. Second, the local features are invariant to location, meaning that it is not important where in the signal the feature appears. From those two properties four concepts can be derived:

- **Sparse connectivity**, which means that not every output unit of a layer is connected to every input unit, which accounts for the highly correlated local features.
- **Parameter sharing** is the idea that each element of the kernel is used at every input position. This corresponds to the location invariance of local features [35].
- **Pooling** which replaces the output by a local summary to gain invariance to small translations [27].
- **The use of a deep architecture** is supported by the property of natural signals which are often composed in hierarchies in which high-level features are formed by a combination of low-level features.

The implementation of sparse connectivity and parameter sharing form a convolution operation, hence the name convolutional neural network [35].

Those four key concepts are described in the following.

3.2.1 Sparse connectivity

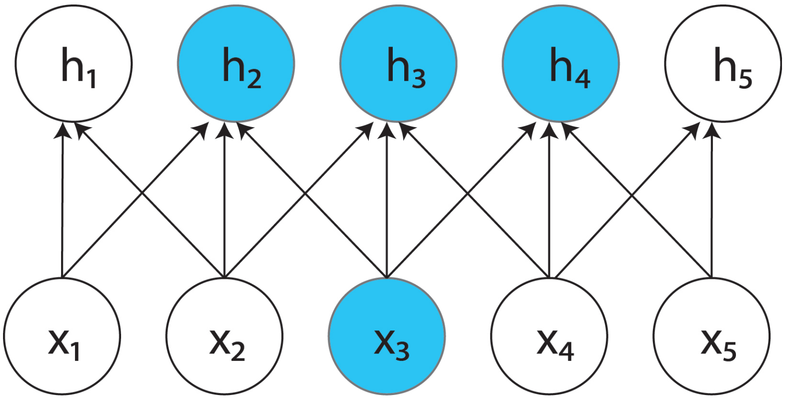
Sparse connectivity or local receptive field is achieved by making the kernel smaller than the input. As a result, not every output unit is connected to every input unit [27]. This is an important distinction to the multilayer perceptron discussed in Section 2.2.3. For a deeper understanding of sparse connectivity see Figure 3.1 for a 1D example. The example is using a kernel with width 3. Viewed from the input it can be seen in (a), that if we focus on one of the input units x_3, three output units h_2, h_3, h_4 are connected to this output. In (b) the comparison to a fully connected layer is shown, where if we focus on x_3 again, all output units are influenced by this input unit.

See Figure 3.2 for the an extended example with a hidden layer. Now the view from the output is discussed. If we focus on one output unit o_3, (a) shows three hidden units h_2, h_3, h_4 are connected directly and affect this unit with sparse connectivity. All input units $x_1 - x_5$ affect the output unit o_3. Those input units that affect the output are called the *receptive field* of o_3 [27].

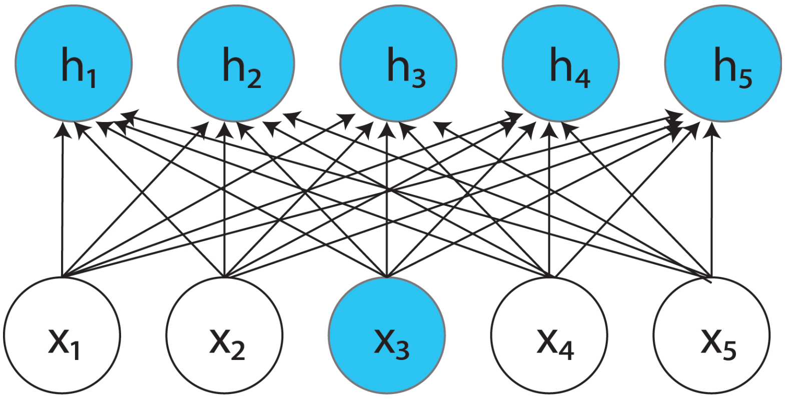
For an example of sparse connectivity applied to a 2D image see Figure 3.3.

3.2.2 Parameter sharing

When using CNNs parameter sharing is applied, which means that the same kernel parameters are used at every position of the input. The idea is that if a set of parameters is useful in one location, it is likely that it will be useful in another location. Instead of every neuron in a layer learning its own parameters, the same parameters are used by every neuron in one feature map. Each feature map has its own set of parameters. Parameter sharing is depicted in Figure 3.4.



(a) Sparse connectivity



(b) Full connectivity

Figure 3.1: (a) Sparse connectivity: An input unit x_3 with a receiptive field of 3 is only connected to 3 output units. (b) Fully connected: x_3 is connected to every output unit. Image adapted from [27].

Parameter sharing and sparse connectivity leads to a significant reduction of parameters that needs to be stored in memory [27].

Consider an example of a small 32x32x3 image. If a feature map of 32x32 should be created in a fully connected way this would lead to a total of 32x32x3x32x32 = 3.14 million weights to train. With a convolutional layer with a receptive field of 3x3 and parameter sharing the number of weights reduces to 3x3x3 = 27 weights for each feature map that should be created in the next layer. From this example it can be seen that the network with convolutional layers needs less memory, is computationally more efficient and thus easier to train.

3.2.3 Pooling

Pooling is used to reduce the amount of parameters in the network. This improves computational and statistical efficiency, and also reduce the memory required [27]. The

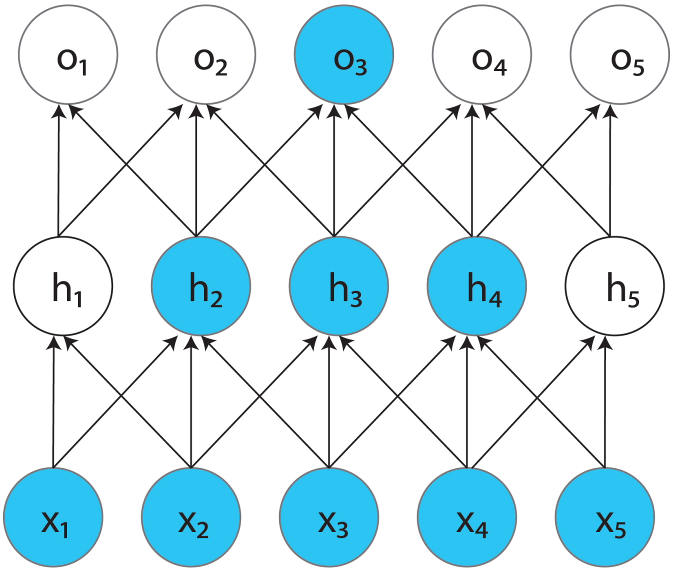


Figure 3.2: Sparse connectivity viewed from above. If we highlight one output unit o_3 the receptive field in the input layer $x_1 - x_5$ is shown. Figure adapted from [27]

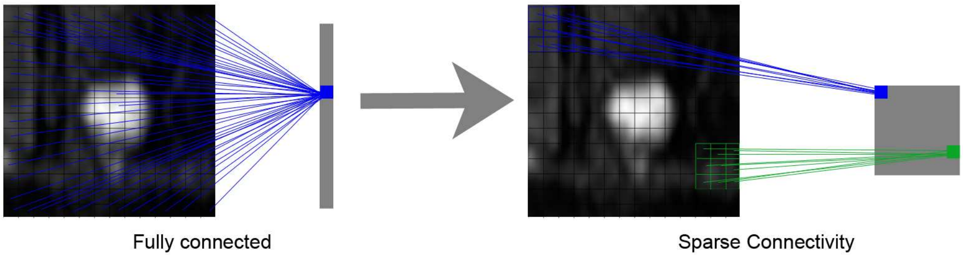


Figure 3.3: *Left:* A fully connected neuron connected to every pixel of the input. *Right:* Sparse connectivity creates a feature map in which every neuron is only connected to neurons of the input in a 3x3 neighbourhood of the neuron.

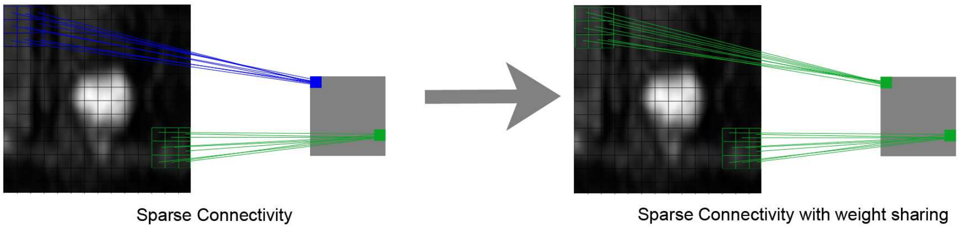


Figure 3.4: Parameters are shared by every neuron in one feature map. *Left:* A feature map connected with sparse connectivity to an input. Green and blue represent a different set of weights learned for the neuron in the feature map. *Right:* A feature map in which the weights are shared within the feature map.

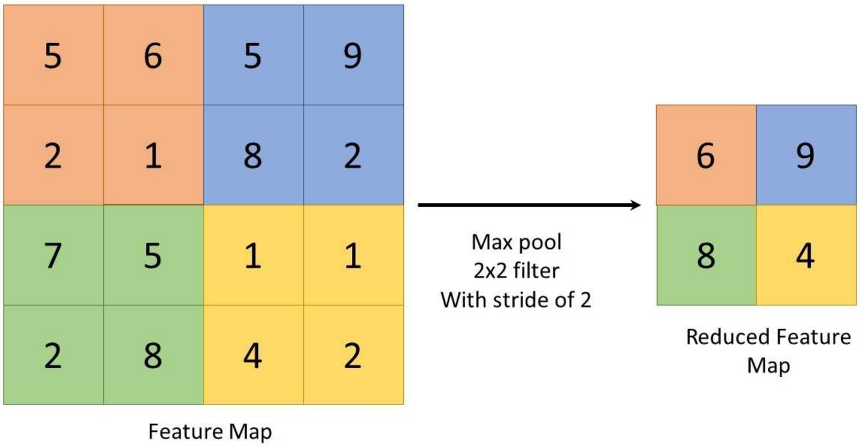


Figure 3.5: Max pooling with stride of 2 and a receptive field of 2x2 on a single feature map. The reduced feature map is reduced by a factor of 2 in width and height.

idea is to compute summary statistics of nearby neurons in a feature map making the representation approximately invariant to small translations in the input [27]. Each depth slice is independently computed. Pooling is combined with down-sampling, that means that the operation reduces the width and height of the feature maps, while the depth remains the same. Max-pooling is the most popular choice in CNN architectures. Figure 3.5 shows the concept of max-pooling. The feature map is parsed with a stride of s in a neighbourhood of size $z\ x\ z$, the maximum value is chosen as a representation for the neighbourhood, this value is passed on to the new feature map. The remaining values are discarded.

3.2.4 Use of a deep architecture

A neural network with one hidden layer is able to represent any function, as stated by the universal approximation theorem, first showed for sigmoid output units [17] and later extended to Multilayer Perceptrons in general [29]. However, it has been shown that in many cases a single hidden layer may be infeasibly large or the training algorithms used are not able to learn the function correctly with only one layer [27]. Evidence that a shallow network is more complicated to train than a deep network is given in [5]. *Ba et al.* show that a shallow network with an adapted, more complex training process can perform as well as a deep network [5].

The principle of using a deep architecture also encodes the properties of natural data learned by the network. This data is often organized in a hierarchy so the first layers learn low-level features and the deeper layers high-level features [35].

Another argument for using a deep architecture is the empirical evidence that greater depth of the network improves the generalization for a variety of tasks [27].

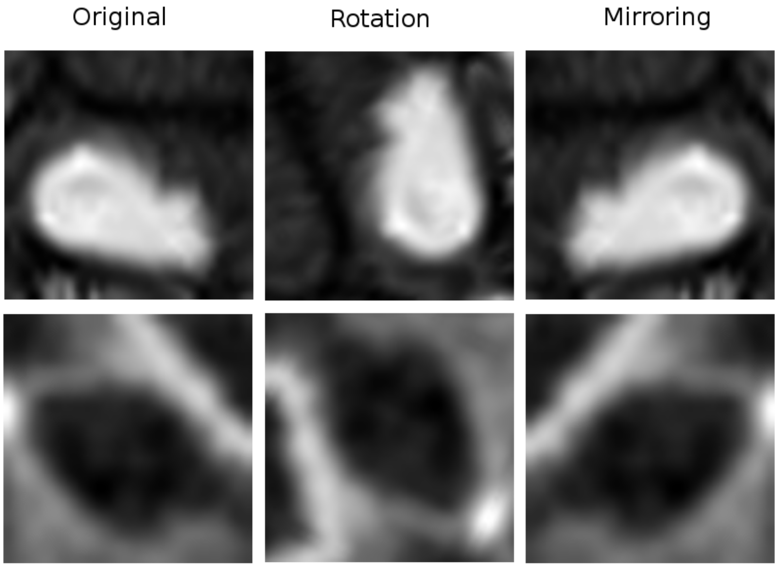


Figure 3.6: Examples for dataset augmentation of bone lesion image patches. The image is rotated randomly and mirrored.

3.3 Regularization Strategies

Due to the use of a high number of neurons in a CNN the capacity of the network is high. With an increased capacity, the risk of overfitting increases as well. To prevent overfitting, different regularization strategies are used for convolutional neural networks. Regularization of a network describes all methods to improve the generalization loss, but not necessarily the training loss [27]. Multiple different regularization stategies were developed, see [27] and [8] for an overview. In the following, only regularization approaches that are used in the course of this thesis are described.

3.3.1 Dataset Augmentation

A simple method to improve the generalization of a network is to add more training data. Since the training data is usually limited, augmentation aims to create meaningful fake data from the existing training data to enrich the dataset. Dataset augmentation is commonly used for image data. An image can be transformed by rotating, mirroring, scaling or translating by a few pixels to create new data. It is important that only transformations that do not change the corresponding label for the sample are applied. Also the classifier must be invariant to the transformations used [27].

Figure 3.6 shows examples for augmented image patches showing a bone lesion.

3.3.2 Dropout

Dropout [65] is a regularization strategy for neural networks. The motivation is that ensembles of networks and the averaging of their outputs usually performs better than using a single network. The challenge of training an ensemble of training multiple deep networks is that it is computationally expensive and tuning the hyperparameters for the networks is daunting [65]. Also, it is not be feasible to run multiple network predictions at test time [27]. Dropout tries to provide an approximation of the predictions of many deep networks. The idea is to randomly "dropout" units in the network, which means all their incoming and outgoing connections are deleted temporarily. A unit is dropped out with a certain probability p during training. This is equivalent to train a thinned version of the network with only units that are not dropped [27]. Each time a training sample or a mini-batch of samples is fed to the network, a new thinned version of the network is created and trained. The weights are shared between all those subnetworks that are created by dropout. This makes it possible to store all subnetworks in an amount of memory as high as storing the full single network. Usually only a fraction of all possible networks are trained explicitly during training [65].

3.3.3 Early Stopping

If a CNN with the capacity to overfit the task is used it can be observed that after a certain number of training epochs the error computed on the training set decreases steadily, while the error computed on an independent validation set begins to rise again [27] [8]. See Figure 3.7 for a graphical explanation of this phenomenon.

In order to achieve a good generalization error, the idea of early stopping is to terminate training at the point where the lowest validation error is reached. This strategy is called *early stopping*. When the lowest validation error is rising and the training error is decreasing, this is a indicator that the network is overfitting on the training data and thus the generalization error will probably rise [27].

3.4 Design of a Convolutional Neural Network

When creating a CNN, a number of design choices determining the architecture and capacity of the network have to be made. These choices are mostly done by setting hyperparameters.

In this section, hyperparameters important for convolutional networks are discussed. As discussed in Section 2.1.4 hyperparameters are parameters that are not learned during training, but are rather set manually before training. In the following, an overview of hyperparameters of CNNs used in the course of this thesis are introduced.

First, parameters determining the networks' architecture are explained. Afterwards, parameters of for governing the training with process stochastic gradient descent are explained.

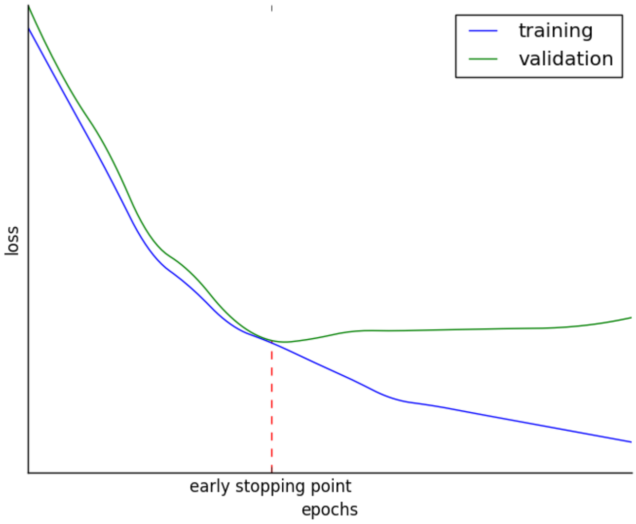


Figure 3.7: Early stopping: While the training error decreases steadily, the validation error begins to rise again. The training can be stopped at that point.

3.4.1 Hyperparameters of the network

Hyperparameters discussed in this section are parameters of the network itself. They set the properties of the network and determine its capacity.

Number of hidden layers and units

The fact that deeper networks generalize better in most cases (see Section 3.2.4) advises to set the number of hidden layers or depth of the network to a high value (typically $>$ 10). Depending on the data used for the network this number varies [27].
The number of hidden units can be set for each layer individually. With an increasing number of hidden units the capacity of the network is increased. If the number of hidden units is too high this usually does not affect generalization [7]. However, both time and memory costs are rising for training and testing the model [27]. If the number of hidden units is too low, the network cannot capture the underlying distribution and will fail to generalize [7].

Convolutional kernel width

For each convolutional layer the kernel width can be determined individually. Increasing the width increases the number of parameters of the network and thus the capacity. On the downside, wide convolutional kernels lead to more memory and storage consumption [27]. Usually a small size of 3x3 or 5x5 is useful, since we want to learn local features of the input.

Neuron non-linearity

The non-linearity of the neurons is given by their activation functions, see Section 2.2.2. As stated there, ReLu is a popular choice for non-linearity of the hidden neurons and the logistic sigmoid or softmax function for output neurons.

Weight and bias initialization

The bias of a neuron can be set to zero at begin of the training [7].
In order to break the symmetry between hidden units in the same layer the weights have to be initalized to a small randomly chosen non-zero value [7].
A popular choice for such an initalization is the Glorot uniform initalization [26]- The weights are initialized according to Equation 3.5.

$$W \sim U[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}] \tag{3.5}$$

where n_j is the size of layer j. With this, n_j can be seen as the input size to a convolutional layer and n_{j+1} as the output size of the layer.

Drop out probability

The probability of drop out influences the capacity of the network. If the drop out is low the capacity of the network is high, but the network might be prone to overfitting. If the drop out is high, the capacity might be too low to fit to the data distribution [27].

3.4.2 Hyperparameters of stochastic gradient descent

The hyperparameters in this section are parameters of the SGD training algorithm. They have to be tuned to ensure a proper training of the network.

Learning rate

Choosing an appropriate learning rate is crucial for a network to train appropriately. An improper value restricts the capacity of the network due to optimization failure [27]. Typical learning rates have values smaller than 1 and larger than 10^{-6} [7]. The learning rate should be optimized when using SGD [7]. Strategies to adapt the learning rate over time can be used [27]. The usefulness of an adaptive learning rate is dependent on the task [7].

Momentum

Setting the momentum is less critical than finding the right learning rate. A value of 0.9 is a common choice in literature [27]. The use of a momentum is explained in Section 2.3.2.

Number of epochs

The number of epochs or training iterations is easy to optimize. With the help of early stopping (see Section 3.3.3), it is possible to stop training when a small generalization error is reached.

Minibatch size

The size of minibatches do not so much impact the generalization error as it impacts the training time [27]. Usually a value between 1 and a few hundred is chosen [7]. With a larger minibatch size it is possible to increase computational speed. However, more epochs are needed because there are less weight updates per epoch [7].

3.5 Convolutional Neural Networks in Medical Imaging

Lo et al. were the first to apply CNNs to medical imaging data [39]. They used a CNN with four layers to detect nodules in X-ray images.

Since then, CNNs were used for a wide variety of tasks including classification, detection, segmentation or registration [38]. An overview of approaches that apply CNNs to medical imaging is given by *Litjens et al.* in [38]. In most applications CNNs are now state-of-the-art.

Here, only those approaches that are most related to the thesis are discussed. Those are either methods focusing on lesion classification or lesion detection. *Setio et al.* use a multistream architecture to classify nodule candidates into nodule or non-nodule in CT images. They generate candidates with the help of CAD system, which are then classified by a CNN [58].

A multiscale approach is used by *Shen et al.* to classify lung nodules. Patches are extracted from thoracic CT screenings at three different scales and three seperate convolutional networks are trained. Each network focuses on a different patch size. The networks are fused to form a final feature vector which is used for classification [60].

A similar multiscale approach is used in [71] where *Wang et al.* apply multiscale CNNs to detect of spinal metastasis in MR scans.

A detection for liver lesions in CT scans with patches extracted at two different scales is shown in [25]. Patches at the border of the liver are extracted seperately and it is shown that a multiclass approach with the labels lesion/internal liver patch/boundary patch can outperform binary classification [25].

Dou et al. detect cerebral microbleeds with 3D convolutional neural networks. They use MR Images to detect bleeds with a high sensitivity of 93% with a low number of 2.7 false positives per scan [20].

A random-view aggregation is used by *Roth et al.* to avoid overfitting and increase training data variability for the detection of lesions in three different applications: sclerotic matastasis, lymph nodes and colonic polyp detection [55]. The random-view aggregation samples a number of N random observations of a region-of-interest by translation, scaling and rotating. These observations are fed to the network individually

and a final classification result is calculated by averaging the probabilities [55].
Bone lesions in multiple myeloma patients are detected by *Xu et al.* in PET/CT scans [74], which is closely related to this thesis, but their work operates on a different modality. They use a novel cascading W-Net architecture to detect lesions [74].

3.6 Summary

In this chapter, convolutional neural networks are introduced. The four main concepts behind convolutional networks: sparse connectivity, parameter sharing, pooling and the use of a deep architecture are described in detail. Regularization strategies for CNNs are introduced and those used in this thesis are discussed. Afterwards, the hyperparameters most relevant to CNNs are described. State-of-the-art applications that are most related to the thesis are described briefly.

CHAPTER 4

State of the Art: Transfer Learning

Transfer learning is a technique in machine learning in which the goal is to transfer knowledge from one task (*source task*) to improve the learning in a different but related task (*target task*) [69].
Transfer learning was first applied for neural networks by *Pratt et al.* in 1991 [50]. Transfer learning can improve the performance of a model if there are few training samples for the target task, but significantly more samples for the source task [27].
In this chapter, a basic introduction to transfer learning is given. A detailed discussion about transfer learning is out of the scope of this work, a survey and in-depth discussion on transfer learning for machine learning and the different applications can be found in [46] and [72].

After introducing transfer learning, the state of the art of transfer learning for CNNs is discussed in more detail. Three different methods of transfer learning with CNNs are introduced: Off-the-shelf use as input for another classifier, only training newly added fully-connected layers in the target domain and fine-tuning. Finally, transfer learning for CNNs used in the medical domain is discussed.

4.1 Basic Concept of Transfer Learning

In a classical supervised learning setting a model is trained and used on the same domain. The assumption made is that the labeled samples used for training and the samples used during application have the same underlying distribution in the same domain [46].
With transfer learning this assumption does not hold. The knowledge gained in one domain in one task can potentially be transferred to another task, potentially in another domain [72]. In Figure 4.1 the basic workflow of transfer learning is depicted. With

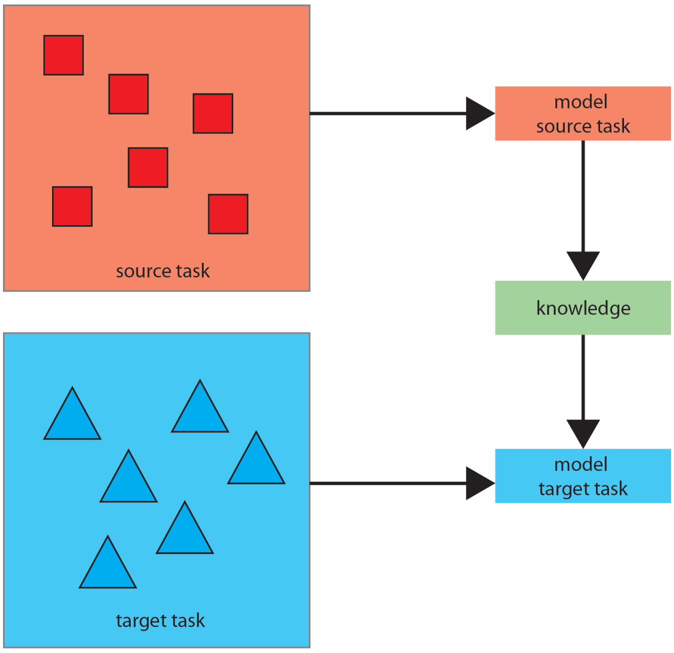


Figure 4.1: Basic concept of Transfer learning. The knowledge learned from the source task is used in combination with samples from the target task to build a model for the target task.

the samples of the source domain one model is trained. Subsequently, knowledge is extracted from this model. This knowledge can have various forms: instances, feature-representations, parameters or relational knowledge can be transferred between models [46]. Afterwards, this knowledge and samples of the target domain are used to improve the learning for the target task.

Transfer learning should be distinguished from multi-task learning [13], which tries to learn several tasks simultaneously. With multi-task learning, the information between the tasks can flow freely and the goal is that for the tasks to support each other during training [69]. With transfer learning the focus is the target task. The information flows from source to target task, and only the performance of the target task is important [46].

4.1.1 Formal Definition

A more formal definition of transfer learning is given in the following. For the formal definition the notation of [46] and [72] is followed.
A domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$ is defined by two components: $\mathcal{X}$ is the feature space and $P(X)$ the marginal probability distribution, with $X = \{x_1, \ldots, x_n\} \in \mathcal{X}$.

Given a domain $\mathcal{D}$ a task $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$ is defined by a label space $\mathcal{Y}$ and a prediction function $f(\cdot)$. The function $f(\cdot)$ predicts a label $y = f(x)$ for a new instance x. It is important to note that the function is not observable directly and is learned from pairs of trainings data $\{x_i, y_i\}, x_i \in \mathcal{X}, y_i \in \mathcal{Y}$.

The function $f(x)$ can be seen as the conditional probability $P(y|x)$ of output y given an input x.
Furthermore, data of the source domain $\mathcal{D}_S$ and the target domain $\mathcal{D}_T$ is needed for transfer learning. Let $D_S = \{(x_{S_1}, y_{S_1}), \ldots, (x_{S_n}, y_{S_n})\}$ be the data of $\mathcal{D}_S$, with $x_{S_i} \in \mathcal{X}_S$ is an instance of the source domain and $y_{S_i} \in \mathcal{Y}_S$ the corresponding label. Similar, we define $D_T = \{(x_{T_1}, y_{T_1}), \ldots, (x_{T_m}, y_{T_m})\}$ as the data of the target domain. $x_{T_i} \in \mathcal{X}_T$ is a target domain instance with its corresponding label $y_{T_i} \in \mathcal{Y}_T$.

For transfer learning either $\mathcal{D}_S \neq \mathcal{D}_T$ and/or $\mathcal{T}_S \neq \mathcal{T}_T$ applies. For a domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$ this implies that either the feature spaces are different ($\mathcal{X}_S \neq \mathcal{X}_T$) or the feature spaces are the same but the marginal probability distributions are different($P(X_S) \neq P(X_T)$, with $X_{S_i} \in \mathcal{X}_S, X_{T_i} \in \mathcal{X}_T$).

Similarly, for a task $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\} = \{\mathcal{Y}, P(Y|X)\}$ that means that $\mathcal{Y}_S \neq \mathcal{Y}_T$ or $P(Y_S|X_S) \neq P(Y_T|X_T)$.

Building on these definitions, a definition of transfer learning can be given [46]:

Definition 4.1.1. Given a source domain $\mathcal{D}_S$ and a target domain $\mathcal{D}_T$ and corresponding learning tasks $\mathcal{T}_S$ and $\mathcal{T}_T$, transfer learning aims to improve the learning of the prediction function $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge gained in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ and/or $\mathcal{T}_S \neq \mathcal{T}_T$.

4.1.2 Potential improvements

Following *Torrey* there are three potential improvements transfer learning can lead to [69]:

- *Higher start:* The initial performance in the target task, without any learning, can be higher than without transfer of knowledge. This is made possible due to the knowledge acquired in the source task which is more applicable to the task than random initialization of the machine learning model.
- *Higher slope:* The amount of time until the model is trained to the target task can be shortened. For neural networks that means that fewer epochs are needed for the network to reach a high accuracy.
- *Higher asymptote:* The final performance at the target task might be higher when transferring knowledge.

Those three improvements are depicted in Figure 4.2.

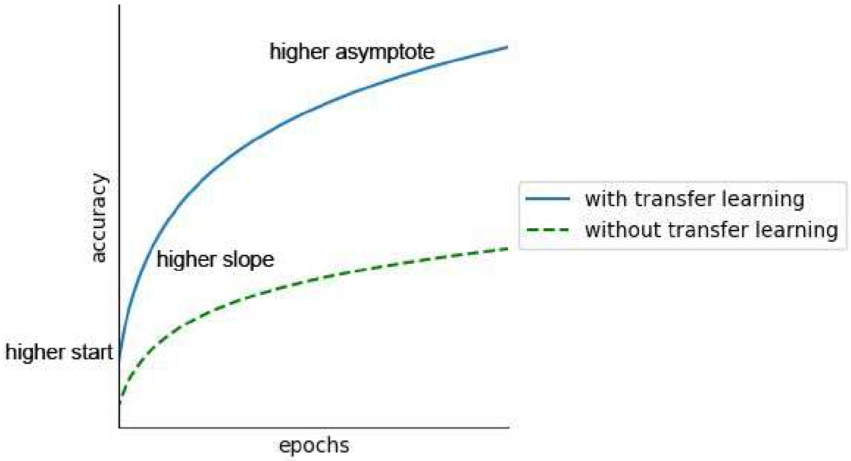


Figure 4.2: Three possible ways of improvement when using transfer learning: higher start, higher slope, higher asymptote. Figure adapted from [69]

4.1.3 When, what and how to transfer?

Three key questions with respect transfer learning are when to transfer, what to transfer and how to transfer knowledge from source to target task [46].

How to transfer is strongly dependend on the machine learning models used. Relevant for this thesis is the question of how transfer can be achieved for convolutional networks, which is discussed in Section 4.2.

When to transfer is rather a question of when not to transfer. When using transfer learning, it is important to avoid negative transfer [46]. Negative transfer occurs when a transfer method decreases the performance on the target task instead of increasing it. A discussion about how to avoid negative transfer can be found in [69].

The most important question is what to transfer. Following *Pan and Yang* transfer learning can be categorized by what is transferred in the following categories:

- *Instance-transfer* can be used when the conditional distribution is similar in the source and target domain [72]. The idea is to reweight instances of the source domain so they can be used in the target domain [46].

- *Feature-representation-transfer* transfers the learned feature representations. Features learned in the source domain are transferred and used in the target domain to improve the performance on the target task [46].

- *Parameter-transfer* approaches share hyperparameters of the source and target learner models. The knowledge transferred is information about how to set the parameters of the target learner model [72].
- *Relational-knowledge transfer* is based on the idea that some relationships among data of the source and target domain are the same. Thus, this knowledge about the relations can be transferred [46].

4.2 Transfer Learning for Convolutional Neural Networks

The knowledge transferred for CNNs are the learned parameters (weights and biases) and in part the network architecture used. In terms of categories discussed in the previous section, transfer learning for CNNs deals with feature-representation-transfer, and depending on the exact approach used, it includes parameter-transfer.

That the transfer of learned parameters is helpful and how transferred parameters can be used is heavily researched. A method to visualize the learned feature maps of a CNN is showed by *Zeiler and Fergus*. With their approach, they are able to show that the first layers in a CNN learn general features which are not dependent on the task, most of them either resemble Gabor filters or color blobs. The deeper layers in the network learn features that are more specific to the task [76]. Figure 4.3 shows the first layer of a CNN. In this visualization it can be seen that the features in the first layers learn basic patterns (e.g. edges) and are thus potentially more useful to transfer to other tasks than deeper layers which are more task specific.

In [75] *Yosinski et al.* analysis how transferable the parameters of CNNs are. They present two main results of their anylsis. (1) Deep layer neurons are more specialized than lower layer neurons, thus they perform worse on the target task without domain adaption. (2) Even if the gap between source and target domain grows, transferring the parameters yields better results than random initialization [75]. This is an important finding for this thesis, because the aim is to transfer features between domains with a large gap (natural images and medical images).

Afridi et al. research if the source task for a given transfer learning problem can be chosen automatically. A method of ranking different source tasks is introduced and it is shown that source tasks related closely are better for transfer learning [3].

In [63] *Soekhoe et al.* investigates the influence of the target task training set on the usefulness of transfer learning. Smaller datasets can benefit from transfer learning more and freezing only the first layers is advisable since they capture general features. On the other hand, the deeper layers can benefit from fine-tuning.

Different approaches on how to proceed the training in the target domain after the knowledge is transferred from the source task were developed. Three approaches can be distinguished

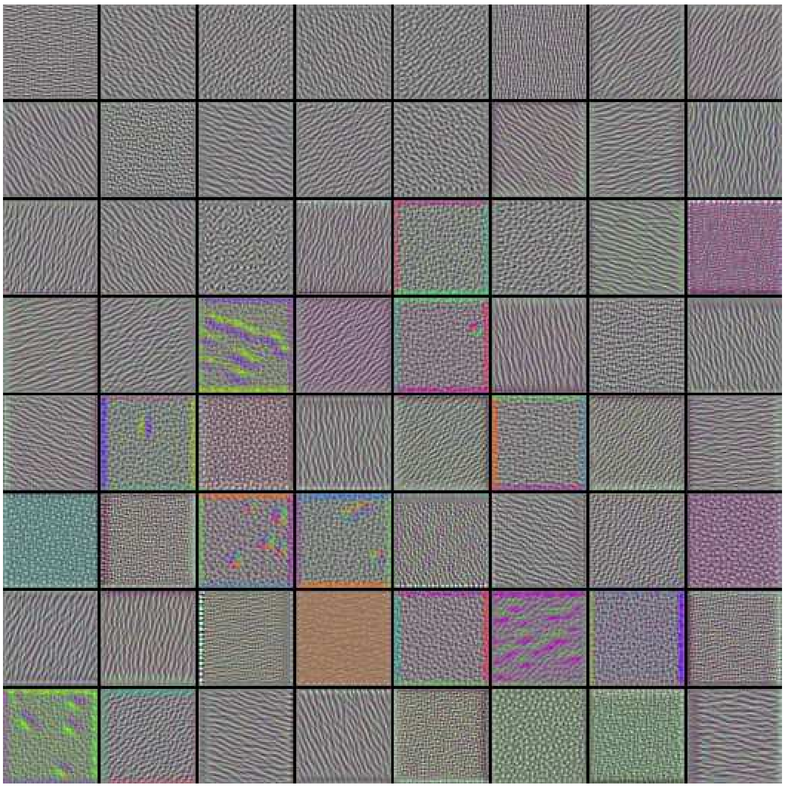


Figure 4.3: Sample features of the first layer of a Convolutional Neural Network

- extracting features from the network to be used in another classifier
- transfer of parameters and training only new classification layers
- Fine-tuning of at least one convolutional layer

The first phase, the training on the source task is the same for all approaches. A large scale dataset is used to train the CNN on the source task. In most approaches, the dataset used is (a subset of) ImageNet [31]. ImageNet is a natural image database with currently over 14 million samples and 20 thousand categories. The samples in the dataset are organized in a semantic hierarchy [31]. For most transfer learning approaches a subset of ImageNet is used. After successful training of the CNN on the source task, the knowledge gained can be transferred. The three approaches, of this transfer are depicted in Figure 4.4 and described in the following.

Pre-train, extract features and use in other classifier (e.g. SVM)

The pre-trained CNN is treated as a feature extractor. The network is used as it is, meaning its parameters are fixed after pre-training on samples of the source domain. Features are extracted from one of the fully-connected layers at the end of the network.

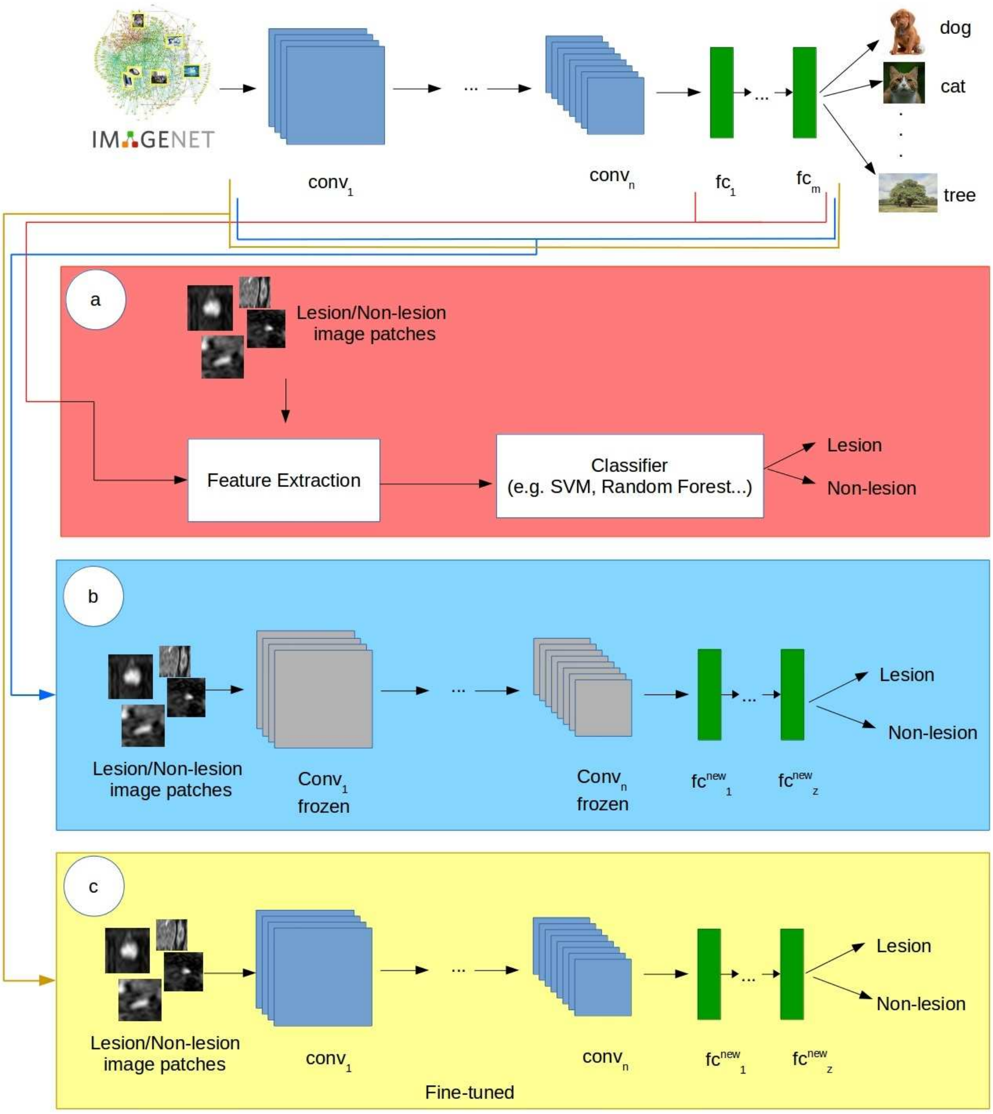


Figure 4.4: Three possible methods to transfer knowledge when using convolutional networks. *(a):* Pre-train, extract the features and use in a classifier (e.g. SVM, Random Forest) to adapt to the target task. *(b):* Exchange the fully-connected layers and only train those on samples of the target domain. *(c):* Exchange the fully-connected layers and fine-tune one to all convolutional layers on samples of the target domain.

Those features are then used in another machine learning model, e.g. Support Vector Machines (SVM) [9] or Random Forest [10]. This method is shown in Figure 4.4 (a).
Donahue et al. experiment with this approach [19]. They show that extracting deep convolutional features and classifying them with a SVM can outperform previous state-of-the-art tasks. A similar study by *Razavian et al.* using a different network architecture confirms and extends the results [51]. They suggest that features obtained from a CNN should be the primary feature extraction strategy in visual recognition tasks.

Pre-train, add new fully connected layers and only train new layers

The second approach is to transfer the weights of the layers to a new network. The new network replaces or adds fully-connected layers and the output layers at the end of the network to fit the target task. Those new layers are initialized randomly and only those layers are trained on the new task. The workflow of this method can be seen in Figure 4.4 (b). In [45] a CNN pre-trained on ImageNet is successfully transferred to object and action classification on the PASCAL VOC dataset with this approach.

Fine-tuning a transferred CNN

Fine-tuning first transfers the weights of the source task network to a new network. In this network, the fully-connected layers and output layers are exchanged to fit the target task. Instead of training only the new layers, at least one of the transferred layers is fine-tuned to the new task. Usually, the first n layers of the network are frozen to avoid overfitting [16] [75]. The number of layers that are fine-tuned can be seen as an additional hyperparamter of a transferred network. Fine-tuning is usually done with a smaller learning-rate than learning from scratch to avoid overfitting. Consider Figure 4.4 (c) for a graphical explanation of this approach.
Yosinski et al. analyse the effect of fine-tuning in comparision to only train the new layers of a transferred network. They find that fine-tuning is beneficial [75]. *Chu et al.* give an overview of in which situations fine-tuning is superior to freezing the transferred weights. They find that if the similarity between source and target task is low, fine-tuning is useful [16].

4.3 Transfer Learning for Convolutional Neural Networks in Medical Imaging

The use of the three approaches of transfer learning discussed before in applications of the medical domain is discussed in this section. Most approaches learn a source task in the domain of natural images and apply the knowledge for a target task working on medical imaging. Even though the domains are quite different and this difference would advise against such a transfer, there is empirical evidence that a transfer is meaningful.

In the following an overview of the three methods of transfer learning for CNNs used on medical imaging is given.

CNN as feature-extractor In [6] the Decaf CNN [19] is used as a feature-extractor. The features are combined with low-level features and used in a linear SVM to detected pathologies in chest radiograph data.
Van Ginneken et al. show that using CNN features off-the-shelf in combination with a linear SVM can be used in pulmonary nodules detection [70].

Only train new layers To the knowledge of the author, there is no medical imaging application that only trains the new layers of a network. One possible reason is the gap between the domain of natural images, in most transfer learning application for CNNs the source domain and the medical domain as the target domain.

Fine-tuning [12] and [15] use fine-tuning a pre-trained network to tasks on mammography images. *Carneiro et al.* use a multiview approach to assess the risk of a patient to develop breast cancer [12]. *Chougrad et al.* classify breast lesions into benign and malignant [15].

Shin et al. compare different learning protocols and network architectures for lymph node detection and interstitial lung disease classification [61]. They show that using transfer learning to transfer knowledge from a natural image task to a medical imaging task is beneficial.

Tajbakhsh et al. show an extensive comparison of full training to fine-tuning on four distinct medical imaging applications [68]. The results show that fine-tuning outperforms full training, especially if there is little training data [68].

The deep network Inception v3 [67] is utlilized in [23] and [28]. Both approaches fine-tune the pre-trained Inception v3 to classify skin cancer [23], respectively to detect diabetic retinopathy [28].

In a similar way, the VGG16 network [62] is used to classify skin cancer with a transfer learning approach [52] [42]. One advantage of using well-known networks as Inception v3 or VGG16 is that the pre-trained weights are publicly available for download and thus can be used directly and only a fine-tuning of the weights is necessary.

In [4] and [32] the performance of fine-tuning is compared to the use of the network as a feature-extractor. *Antony et al.* show such that fine-tuning is superior for the assessment of knee osteoarthritis [4]. *Kim et al.* show that the use of a CNN as a feature extractor outperforms fine-tuning in cytopathology classification [32]. These contradicting results show that a general recommondation to use fine-tuning of the CNN as a feature-extractor can not be given [38].

4.4 Summary

In this chapter the concept of transfer learning and its goal, the transfer of knowledge from a source task to a target task is described. A formal definition is given and the potential improvements of learning with transfer learning are discussed. A categorization of transfer learning is introduced.

Transfer Learning for convolutional networks is described, which is the main idea used in this thesis. Three distinct methods of doing such a transfer are described: Using the CNN for feature extraction, only train new classification layers and fine-tuning. For all of those strategies applications in the medical imaging domain are discussed.

CHAPTER 5

Preliminary Experiments

In this chapter, preliminary experiments that are carried out in order to decide on a final methodology are documented. The experiments are performed to determine the methodology used in the course of the thesis. The first experiment (see Section 5.2) is done to decide on a network design. The second to decide on one of the transfer learning strategies described in Section 4.2.

5.1 Dataset for the preliminary experiments

For the preliminary experiments single channel image patches extracted from CT data as described in Section 6.3.1 are used to test the models. The dataset for the experiments consists of 5938 samples split into training (70%), validation (10%) and test set (20%). The training is done on 2135 positive lesion patches and 2124 negative non-lesion patches. The test set, on which the performance measures of the experiments are computed, consists of 538 positive and 530 negative patches. The validation set is used for early stopping in the trainings process of the CNNs.

5.2 Experiment 1: CNN architecture

In a first experiment, different network architectures are tested. The tested network architectures are described in the following:

- **Simple Network** As a base line a simple convolutional network consisting of only three convolutional layers separated by max pooling layers is tested.
- **U-Net** is an architecture developed for medical image segmantation [53]. The networks tries to capture context as well as local information by a contracting path

	Accuracy	Precision	Recall	F-Score
Simple Network	0.74	0.90	0.54	0.68
U-Net	0.81	0.83	0.78	0.80
VGG-16	**0.89**	**0.87**	**0.91**	**0.89**
Inception v3	0.80	0.81	0.80	0.81

Table 5.1: Results of preliminary experiments on different network architectures.

and a expanding path. Approaches building on the U-Net has shown potential for high performance in medical imaging (e.g. [43] [53]).

- **VGG-16** has a simple architecture using only convolutional layers with a small receptive field of 3x3 and max pooling layers [62]. It has been proven that the network can be succesfully transferred to a task in the medical domain [52] [42].
- **Inception v3** is chosen as a network architecture to test, because there are examples for succesful transfer of the network to the medical domain [23] [28]. Inception v3 leverages a deep architecture with so called inception modules as introduced in [66]. For more information about the Inception v3 network see [67].

The simple network and the U-net are trained from scratch on CT image patches. The VGG-16 and the Inception network are fine-tuned on CT image patches from weights that are pre-trained on ImageNet. For fine-tuning, the first six layers of the VGG-16 network are frozen. Inception v3 is a deeper network, and thus the first thirty layers are frozen for fine-tuning.
Stochastic gradient descent with mini-batches and momentum is used to train all networks.

Results

Results of this experiment are shown in Table 5.1. The simple network is too shallow to fit to the task and thus the recall drops. When examining the learning process of Inception v3 it is shown that the network starts to overfit on the training data. Because of the deep architecture of the network, the capacity is also high and overfitting occurs when fine-tuned on a small dataset.

VGG-16 significantly outperforms all other methods in this preliminary experiment. It is chosen as base for the network architecture in the course of the thesis.

5.3 Experiment 2: Transfer learning strategy

A second experiment is done to determine the transfer learning strategy. As described in Section 4.2 three strategies are possible: CNN as feature extractor, only train the new fully-connected layers or fine-tuning the network to the target task.

	Accuracy	Precision	Recall	F-Score
VGG-16 + SVM	0.81	0.81	0.81	0.81
VGG-16 + Random Forest	0.80	0.83	0.75	0.79
Fine-tuning	**0.89**	**0.87**	**0.91**	**0.89**

Table 5.2: Results of preliminary experiments on different transfer learning strategies.

Only training fully-connected layers is, to the knowledge of the author, not used in any medical imaging application and thus it is not evaluated in this experiment.
The strategy of using a CNN as a feature extractor is tested against fine-tuning. All approaches are tested with a pre-trained VGG-16 network. The network is pre-trained on natural images. The feature extracting approach is used with two different classifiers: Support Vector Machines (SVM) and Random Forests.

A SVM [9] is a classifier that discriminates between two classes with a hyperplane which divides the input space. This is done by an algorithm that tries to find a plane that separates samples belonging to class A from those belonging to class B. More information about Support Vector Machines can be found in [8].

Random forest [10] is an ensemble learning method, which learns multiple decision trees at training time. The prediction of the forest is a majority vote of those trees [59]. For a deeper introduction to random forests see [10].

For the implementation of the SVM and the random forest approach the software package *Scikit-learn* [47] is used. For the random forests classifier 30 estimators are used. Features that are fed to the SVM and the random forests are extracted from the first dense layer of the VGG-16 network.

Results

Results of the experiments are shown in Table 5.2. Fine-tuning the VGG-16 clearly outperforms the feature extraction approaches and is used as transfer learning strategy in the methodology of this thesis. A possible reason for this is, that when transferring knowledge from a source domain that is different from the target domain fine-tuning the network is beneficial.

5.4 Summary

In this section, two preliminary experiments to decide on a methodology used for this thesis are described. As a result of these experiments the CNN architecture is chosen to follow those of the VGG-16 network and the transfer learning strategy is fixed to fine-tuning the network. With these results a final methodology used in this thesis can be described.

CHAPTER 6

Methodology

In this chapter, the methodology of the thesis is described. The methodology builds on the state-of-the-art discussed in the chapters up to now. First, bone masking used to limit the region-of-interest of the methodology is explained. Next, the architecture of the CNN used is shown and it is argued why choosing an architecture already in place is beneficial for transfer learning. Afterwards two different strategies for patch extraction are introduced. One extracts single channel patches, the second tries to make use of the three channel input layer of the used CNN architecture by creating three channel patches where each channel focuses on different information. Next, the two learning protocol transfer learning with fine-tuning and random initialization are described. Afterwards, the models trained for the thesis are discussed. Finally, two ways to use the trained models are shown: classifying new imaging data and the use of a volume parsing approach. It is described how to parse whole body scans with this method and how the output of this method can be interpreted.

6.1 Bone masking

For each CT/MRI volume used a corresponding bone mask has to be provided for the methodology developed. Bone masking is needed to restrict the region-of-interest of the methodology. The patch extraction methods (see Section 6.3), as well as the method for volume parsing (see Section 6.7) rely on the bone mask to perform correctly. The bone mask has to be provided in the dataset, the automatic generation of the bone mask is out of the scope of this work.

6.2 Convolutional Neural Network - Architecture

The architecture for the CNN used is derived from the VGG-16 architecture. The choice of architecture is argued in Section 5.2. The VGG-16 architecture is developed to

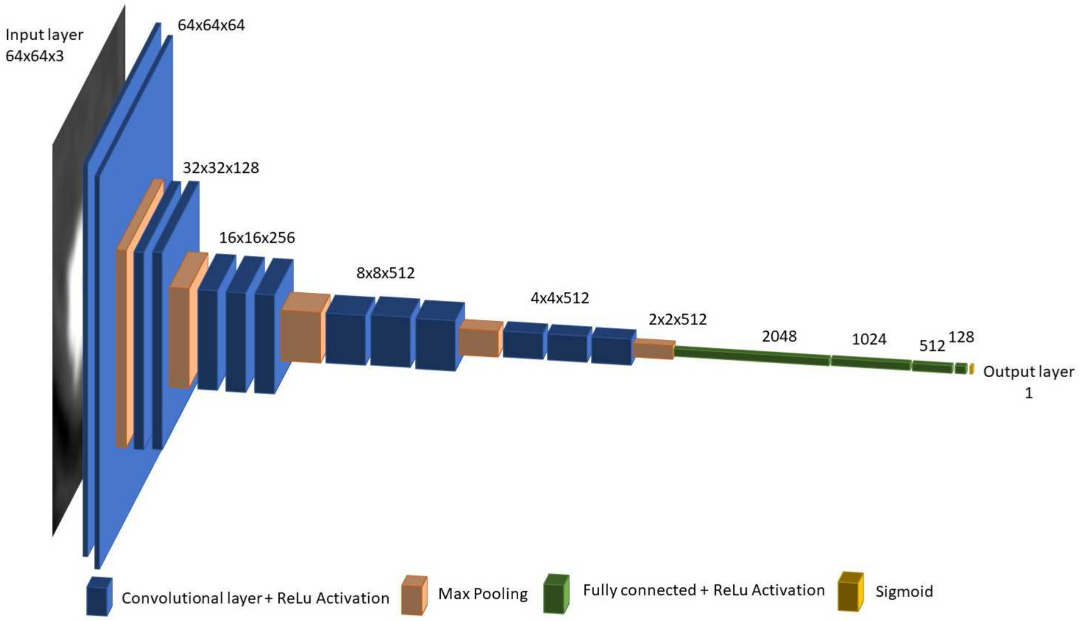


Figure 6.1: The Convolutional Neural Network, as used in the experiments. It is based on VGG-16 [62].

classify natural images in the ImageNet Challenge 2014 [62]. The idea is to use small convolutional filters of size 3x3 and using a deep architecture with 16 weight layers to improve the performance.
The VGG-16 architecture is adapted as shown in Figure 6.1. Instead of using inputs of shape 244x244x3, a input layer with a reduced spatial resolution of 64x64x3 is used. The architecture of the hidden layers remains the same as in the original VGG-16 architecture. Five convolutional blocks with two, respectively three convolutional layers are used. All convolutional layers have a receptive field of 3x3 and use a ReLU as activation function. The convolutional blocks are separated by max pooling layers with a receptive field of 2x2. After the first three pooling layers, the number of feature maps are doubled starting with 64 feature maps per convolutional layer to finally 512 in the later convolutional blocks. The fully-connected layers at the end of the original network are replaced in order to fit the detection task and to produce a single value output. The output function is a sigmoid function to produce a probability value of seeing a lesion.

Due to the popularity of VGG16 in many transfer learning approaches (e.g. [40], [42], [52]) choosing a VGG16 architecture has the advantage that pre-trained weights exist that can be used directly. The weights used are extracted from a VGG16 training on natural image classification using the ImageNet dataset. This has the advantage that a training on the source task by this implementation is not necessary. Only the weights are copied.

6.3 Patch Extraction Strategies

To train a convolutional neural network image patches are needed. Those image patches are extracted from CT and MRI volumes respectively. The strategies for patch extraction remain the same for both modalities. In the volumes bone lesions are annotated and an organ mask, including a segmentation for bones is provided.

For both modalities two datasets are created - one by using a single channel patch extraction and one by using a three channel patch extraction approach.
The two methods of patch extractions are described in the following.

6.3.1 Single Channel Patches

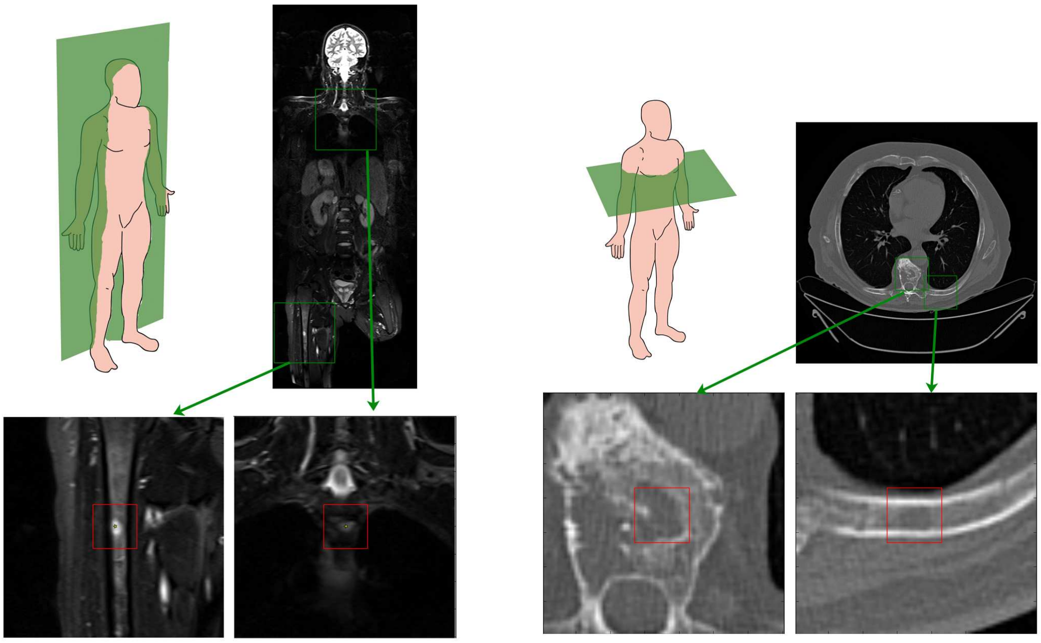
A set of single channel patches $\mathbf{P}^{1C}$ is extracted from whole body volumes $\{\mathbf{V}_1, \ldots, \mathbf{V}_n\}$. For each volume $\mathbf{V}_i \in \{\mathbf{V}_1, \ldots, \mathbf{V}_n\}$ the center position of all lesions are denoted as $\{\mathbf{x}_1^{p_i}, \ldots, \mathbf{x}_j^{p_i}\}$. The bone mask of a $\mathbf{V}_i$ is called $\mathbf{M}_i$. The patches are extracted with a fixed field of view of sxs along a specific axis. For each $\mathbf{V}_i$ and each lesion position $\mathbf{x}_m^{p_i}$ a positive patch $\mathbf{p}_m^{p_i} \in \mathbb{R}^{sxs}$ is extracted centred at $\mathbf{x}_m^{p_i}$. Each patch $\mathbf{p}_m^{p_i}$ is augmented by random rotations and mirroring. This results in a set of positive patches $\mathbf{P}_{p_i}$ for each $\mathbf{V}_i$. A set of negative patches $\mathbf{P}_{n_i} = \{\mathbf{p}_1^{n_i}, \ldots, \mathbf{p}_m^{n_i}\}$ is extracted from $\mathbf{V}_i$ at m random positions $\mathbf{x}_m^{n_i}$ inside $\mathbf{M}_i$, with the restriction that they do not overlap with the extracted patches in $\mathbf{P}_{p_i}$. The negative patches $\mathbf{p}_m^{n_i}$ are not augmented. The final set of patches used for training and testing is given by $\mathbf{P} = \{\mathbf{P}_{p_i} \cup \mathbf{P}_{n_i}\}$.

Figure 6.2 shows the principle extracting positive samples in MRI and CT slices.

For CT volumes the size s, with which the patches are extracted is set to 15 millimeters and the patches are extracted along the axial axes. For MRI volumes, the patches are extracted with s of 40 millimeters along the coronal axis. The different axes along which the patches are extracted are due to the different scanning protocol of CT and MRI. CT has isotrophic distances in the axial planes but a larger slice distance between the axial planes. The extraction follows this protocol. MRI has large slice distances between coronal planes and isotrophic distances within a coronal plane, that is the reason for extraction along the coronal axis.

6.3.2 Three Channel Patches

A set of three channel patches $\mathbf{P}^{3C}$ is extracted from each volume $\mathbf{V}_i$. Positive patches are extracted around the center of a lesion and augmented as described in single channel patch extraction. The same way negative patches are extracted at m random positions inside of $\mathbf{M}_i$. As with the single channel patches the size of patches is 15x15 millimeter for CT volumes and 40x40 millimeter for MRI volumes. The difference is that patches with three channels are extracted. Different density windows are encoded into the channels. The idea is that by splitting the information provided in the volume data into the three channels, the network can potentially find more meaningful features compared to

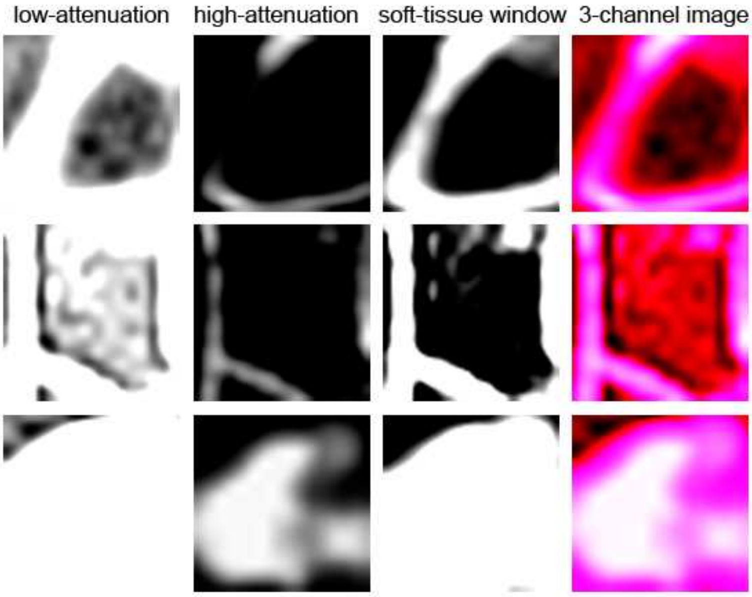


(a) Patch extraction in a MR slice (b) Patch extraction in a CT slice

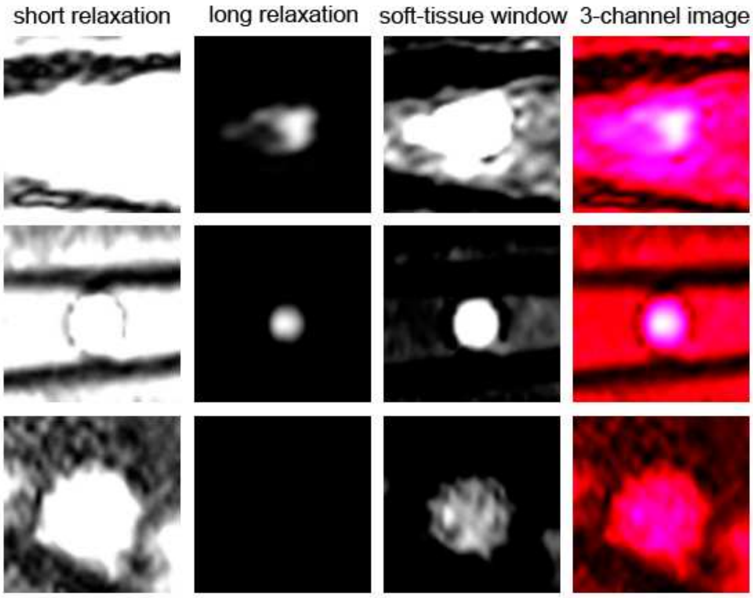
Figure 6.2: Examples of patch extraction of positive samples. The upper row shows the context of the extracted patch. The red border shows the outline of the final patch.

single channel patches, where each input channel receives the same information. The radiodensity in CT data is measured in Hounsfield Units (HU). For CT image patches the first channel focuses on a low-attenuation window of values smaller than 100 HU, the second on a high-attenuation window bigger than 400 HU, and the third on a soft tissue window between 100 and 400 HU. An example of the three channels extracted and the combined image patch is shown in Figure 6.3 (a). From the first two rows it can be seen that the low attenuation window separates the lesion, while high-attenuation and soft tissue window encode context information. The third row shows a lesion with a different visual appearance. The low-attenuation and soft tissue window contain hardly any useful information, while the high-attenuation window segments the lesion.

In MRI, the first channel focuses on a short relaxation time of values smaller than 40 ms, the second channel on long relaxation times bigger than 100 ms. Finally, the third channel encodes a relaxation time between 40 and 100 ms. Examples of the three channels extracted are shown in Figure 6.3 (b). As we see in the CT samples before, lesions can have a wide variety of different appearances. In the first two rows of the figure in the long relaxation and to some extent the soft tissue window as well the lesion is clearly visible. The short relaxation window show the context of the lesion. In the third sample, the long relaxation window shows no information at all, which can be problematic during training.



(a) In CT the three channel patch extraction encodes a low-attenuation, a high-attenuation and a soft tissue window into the first, second and third channel respectively. The combined image patch is shown on right.



(b) In MRI the three channel patch extraction encodes a short relaxation time, a long relaxation time and a soft tissue window into the first, second and third channel respectively.

Figure 6.3: Examples for three channel patch extraction in CT- and MRI data.

From the samples in Figure 6.3 one can see that lesions can appear very differently and a general detection is complicated. Separating lesions only based on their intensity values is not possible.

6.4 Learning Protocols

Two different learning protocols are used: Transfer learning with fine-tuning on lesion/non-lesion patches ($\mathbf{P}^{1C}$ or $\mathbf{P}^{3C}$) and random initialization with training only on lesion/non-lesion patches ($\mathbf{P}^{1C}$ or $\mathbf{P}^{3C}$).

6.4.1 Transfer Learning

From the three possible approaches discussed in Section 4.2, fine-tuning is used. Preliminary experiments with methods using CNNs as feature extractor and then using a Support Vector Machine or Random Forest showed that their performance is inferior to the fine-tuning approach. Details for this experiments can be found in Section 5.3.

An overview of the fine-tuning approach is given in Figure 6.4. The process of fine-tuning consists of training two models.

1. In a first step, the VGG16 is trained on a subset of natural image dataset ImageNet, resulting in a network suitable for multiclass classification of natural images. From this network the parameters (weights and biases of the convolutional layers are transferred to a new model.
2. The second model is adapted to fit the task of computing a probability for a lesion. The fully connected layers and the output function are exchanged. A sigmoid function is used to compute a single probability value. The new layers are initialized randomly and dropout is used as a regularization strategy between these layers to avoid overfitting. The first two convolutional blocks are frozen, because, as argued in Section 4.2, the first convolutional layers usually learn to extract generic features, e.g. edges or color blobs. Fine-tuning is done on the medical image patches ($\mathbf{P}^{1C}$ / $\mathbf{P}^{3C}$). The network is fine-tuned with stochastic gradient descent using early stopping. Binary-crossentropy is used as a cost function.

This results in a final model that takes new lesion patches as input and computes a probability value of seeing a lesion as output. This probability values can be used to classify new imaging data (Section 6.6) or in the volume parsing approach (Section 6.7).

6.4.2 Random Initialization

As discussed in Section 3.4.1, a convolutional network needs an initialization of its weights and biases. The biases are usually set to zero. For the initialization of the weights a Glorot uniform initializer is used [26]. The weight initialization is explained in Section

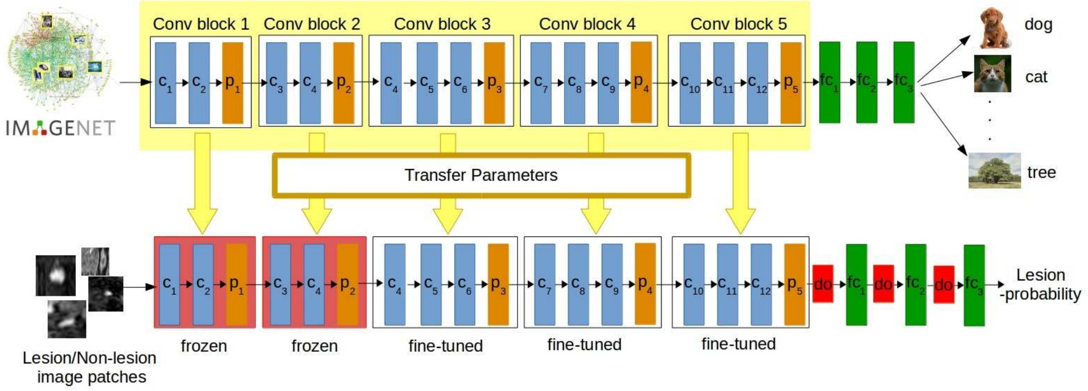


Figure 6.4: Transfer Learning of the VGG-16 network. First, the VGG16 is trained on the natural image database ImageNet. Then the parameters are transferred. The first two convolutional blocks are frozen and new classification layers are added at the end of the network. Then, the network is fine tuned on lesion image patches.

		Patch extraction	
		Single channel	Three channels
CT	Transfer Learning	$\mathbf{TL}_{CT}^{1C}$	$\mathbf{TL}_{CT}^{3C}$
	Random Initialization	$\mathbf{RI}_{CT}^{1C}$	$\mathbf{RI}_{CT}^{3C}$
MRT	Transfer Learning	$\mathbf{TL}_{MR}^{1C}$	$\mathbf{TL}_{MR}^{3C}$
	Random Initialization	$\mathbf{RI}_{MR}^{1C}$	$\mathbf{RI}_{MR}^{3C}$

Table 6.1: Overview of the models trained. For each modality, both patch extraction strategies paired with both learning protocols are trained.

3.4.1.
From this initialization the network is trained only on the medical image patches extracted as described in Section 6.3. The training is done with stochastic gradient descent with momentum. Early stopping is used to determine the training when the network starts to overfit on the training data. Binary-crossentropy is used as cost function. Drop out is used as a regularization approach between the fully connected layers of the network.

6.5 Model training

All combinations of patch extraction strategies and learning protocols are trained on both modalities. This accounts for a total of 8 models trained, see Table 6.1 for an overview.

The models will be denoted with the following name scheme: $\mathbf{MO}_{LP}^{PE}$, where MO is the Modality either CT or MR. LP is the learning protocol, either TL for transfer learning

or RI for random initialization. Finally, PE is the patch extraction strategy 1C stands for single channel patches, 3C for three channel patches.
These models are compared and discussed in course of the thesis. Their strength and weaknesses are analyzed.

6.6 Classifying new imaging data

After the models are trained, they can be used to classify new imaging data. A patch extracted from CT- or MRI data with the patch extraction strategy corresponding to the patches used during training the model can be fed into the network. The output is a probability value p of seeing a lesion. To get a final class label lesion/non-lesion a simple thresholding is done:

$$label = \begin{cases} \text{lesion if } p > 0.5 \\ \text{non-lesion otherwise} \end{cases} \tag{6.1}$$

With this classification the accuracy of the models can be computed.

6.7 Volume parsing

Another possible use of the models trained is the detection of lesions in previously unseen whole body volumes. For this purpose a sliding window approach is used. The details for this approach are given in this section.
For a volume $\mathbf{V}_i$ the output of the approach is a probability map $\mathbf{O}_i$ of the same shape as $\mathbf{V}_i$.

Algorithm 6.1: Volume parsing

```
Input: $V$, volume to parse
Input: $M$, bone mask corresponding to $V$. If $M[x] == 1$ inside bone, $M[x] == 0$
       outside bone
Input: $N$, trained CNN used for classification of image patch
Input: $s$, scale of the image patch to extract
Output: $O$, propability map for the volume $V$
for $x \in V$ do
    if $M[x] == 1$ then
        Cut image patch $p$ of size $s$x$s$ around $x$
        Feed $p$ to $N$, store output to $O[x]$
    end
end
```

The basic algorithm is given in Algorithm 6.1. The idea is to sample an image patch at every location of the volume $\mathbf{V}_i$ within the corresponding bone mask $\mathbf{M}_i$. Those image patches are fed to the network and a probability value is computed. This probability value is stored to the probability map $\mathbf{O}_i$ at the corresponding position. The result is a probability map of the same shape as the volume parsed. In this probability map positions with high probability value indicates locations where the network detected a lesion. Corresponding to Section 6.6, only probability values greater than 0.5 indicate the detection of a lesion. All smaller values can be interpreted as non-lesion.

Depending on the modality used and the network used the patches are extracted following Section 6.3. Patches in MRI volumes are extracted with a size of 40x40 millimeter along the coronal axis, CT patches with a size of 15x15 millimeter along the axial axis.

6.8 Summary

In this chapter the methodological approach of the thesis is explained. The architecture of the network which is derived from the VGG-16 network is described in detail. Two patch extraction strategies are introduced. One is dealing with single channel image patches, the other with three channel patches in order to focus on different information in each of the channels. Two learning protocols, random initialization and fine-tuning are discussed. Random initialization is used as a base line to study the effect of transfer learning. The denotations of the eight models that are trained for this thesis are explained. Finally, the use of the trained models to classify new imaging data and for use in the volume parsing approach is described.

CHAPTER 7

Evaluation

In this chapter the evaluation of the approaches compared in this thesis is described. First, the dataset used for evaluation is discussed. Afterwards, performance measures for the quantitative analysis of image patches are described. Next, the method to do the qualitative evaluation of volume parsing whole body scans is introduced. Finally, the setup for the evaluation is described and the results of preliminary experiments on how to set hyperparameters for the evaluation are given.

7.1 Data

The data used for the evaluation of the eight models trained is a subset of the VISCERAL Detection Gold Corpus [33]. The corpus contains 50 whole body CT scans and 50 whole body MRI-T2 scans. Out of those 50 CT and 50 MRI-T2 scans, organ masks exist for 25 CT respectively MRI scans. Because of the methodology described in Chapter 6 only scans with corresponding masks are used for the evaluation.

Out of the 25 scans of each modality, three are used as test volumes for volume parsing. The other 22 are used to train and evaluate the CNNs on image patches. The image patches extracted are randomly split into a training set (72%), a validation set (10%) and a test set (18%). The CT volumes have voxel dimensions of [0.78-0.98 x 0.78-0.98 x 1.5] millimeters. The MRI scans have a spacing of [0.78-1.3 x 6 x 0.78-1.3] millimeters.

Lesion annotations are given by a centre point of the lesion and two additional points on the perimeter. The additional points give an estimate of the size of the lesion. Lesions are annotated in bones, brain, liver, lungs and lymph nodes. In this thesis only bone lesions are used.

Examples for lesion annotations are shown in Figure 7.1.

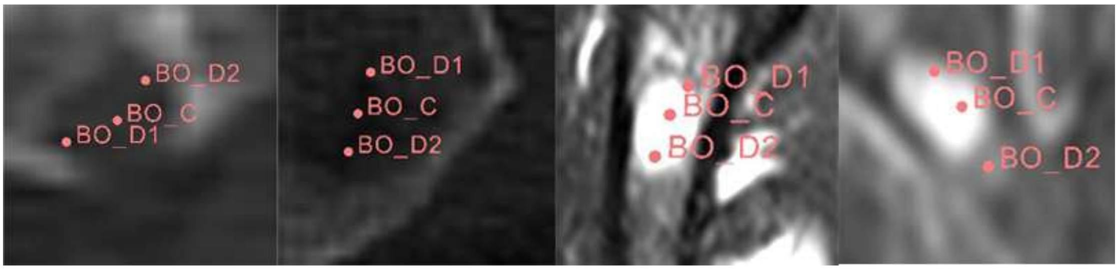


Figure 7.1: Samples for lesion annotations. Lesions are defined by a center point (C) and two points corresponding to the perimeter (D1 and D2). Left: Two samples in CT slices. Right: Two samples in MR slices

	lesion	non-lesion
Training set	2153	2124
Validation set	299	294
Test set	538	530
	2990	2948

Table 7.1: Number of samples extracted from CT volumes

7.1.1 CT Image Patches

In the 22 CT volumes, a total of 598 lesions are annotated. 2D image patches are extracted along the axial axis. Positive patches are augmented by mirroring along the horizontal as well as the vertical axis and by two random rotations of the patch. The number of negative samples drawn is set to balance the dataset containing approximately as many lesion as non-lesion patches.
Table 7.1 gives an overview of absolute numbers of the splits for CT volumes after extracting and data augmentation.

With data augmentation a total of 2990 lesion- and 2948 non-lesion patches are extracted. The networks are trained on 2153 lesion- and 2124 non-lesion patches. The validation set is used for early stopping during the training of the models. The performance measures for CT image patches are computed on a test set of 538 lesion- and 530 non-lesion patches.

7.1.2 MR Image Patches

The 22 MR volumes contain a total of 328 lesions. Image patches are extracted in 2D along the coronal axis. Positive patches are augmented by mirroring along the horizontal and the vertical axis and by three random rotations of the patch. The absolute number of training, validation and test set after extracting and data augmentation are given in 7.2.

The MRI patches dataset consists of a total of 1968 lesion- and 2200 non-lesion patches.

	lesion	non-lesion
Training set	1418	1584
Validation set	196	220
Test set	354	396
	1968	2200

Table 7.2: Number of samples extracted from MRT volumes

The models are trained on 1418 lesion patches and 1584 non-lesion patches. The test set used to compute the performance measures consists of 354 lesion- and 196 non-lesion patches.

Bone masking

The masks provided for the MR volumes are not fitted to the bone exactly. This makes it difficult to use them directly for the patch sampling method (Section 6.3), as well as the volume parsing approach (Section 6.7). Figure 7.2 depicts problematic cases of the bone mask. There are volumes in which bone lesions are annotated outside of the mask, which should not be the case, since they would stay undetected by the volume parsing method.

For the evaluation of the approach the bone mask is expanded by dilation. See Figure 7.2 for examples of bone masks before and after dilation. This approach is only a heuristic given the visual impression of the bone mask. It does not provide an exact solution to the problem. An exact solution is out of scope of this thesis. Nevertheless, for the patch extraction the approach is sufficient. For evaluating the probability masks the fact of the extended bone mask should be kept in mind in order to interpret potential false positives yielded by this approach. Through the extension the bone mask may include other anatomical structures, e.g. inner organs like liver or intestines which might lead to an increase of false positives.

7.2 Evaluation of classification accuracy on image patches

The accuracy of the classification on image patches is evaluated quantitatively. Different performance measures are used to evaluate the performance of the method. The measures used are commonly used for performance evaluation of classification tasks [64]. The following measures are defined in terms of the contingency table shown in Table 7.3.

Accuracy measures the overall performance of a classifier. It measures the proportion of correctly classified samples to the total set.

$$Accuracy = \frac{tp + tn}{tp + fn + fp + tn} \tag{7.1}$$

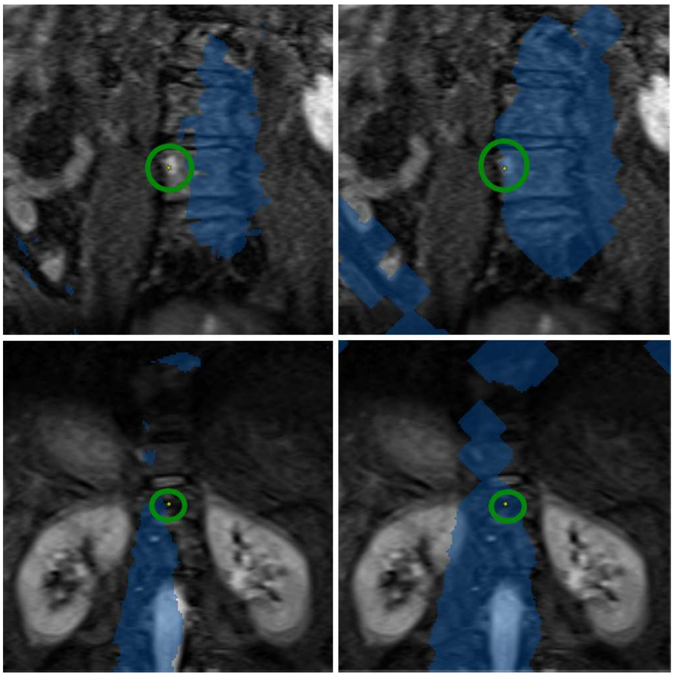


Figure 7.2: Two samples for unexact mask in MR data. The mask is depicted in blue; the lesion is circled in green. *Left:* Before dilation, the annotated bone lesions lie outside of the mask. *Right:* After dilation, the lesion now is inside of the mask.

Precision is a measure for how many of the lesions detected by the system are annotated as lesions in the groundtruth.

$$Precision = P = \frac{tp}{tp + fp} \tag{7.2}$$

Recall measures how well the classifier can identify lesions:

$$Recall = R = \frac{tp}{tp + fn} \tag{7.3}$$

F-Score is the harmonic mean of precision and recall and is used as a single value for the overall performance of the classifier:

$$F = \frac{2}{\frac{1}{R} + \frac{1}{P}} \tag{7.4}$$

Receiver Operating Characteristics (ROC) Curve The Receiver Operating Characteristics (ROC) Curve and the associated Area under the curve (AUC) is another performance measure used to evaluate classification algorithms. ROC graphs plot the

		Groundtruth	
		lesion	non-lesion
Prediction	lesion	true positive (tp)	false positive (fp)
	non-lesion	false negative (fn)	true negative (tn)

Table 7.3: Contingency table for classification in lesion and non-lesion.

sensitivity against the false positive rate (1-specificity) [24]. Sensitivity or true positive rate (TPR) is another word for recall as defined in Equation 7.3. The false positive rate (FPR) is defined as:

$$false\ positive\ rate = 1 - specificity = \frac{fp}{fp + tn} \tag{7.5}$$

The line between (0,0) and (1,1) of a ROC curve represents a classifier randomly choosing a sample to be negative or positive. The point (0,1) represents perfect classification. The area under the curve (AUC) is a measure for the probability that a randomly sampled positive instance higher than a randomly sampled negative instance. Because of the line between (0,0) and (1,1) a classifier should never have an AUC under 0.5 [24].

7.3 Evaluation of volume parsing

During the evaluation of volume parsing the aim is to assess the ability of detecting lesions in previously unseen whole body volumes. Due to the nature of the data used for evaluation, the performance of the volume parsing approach on whole body volume is evaluated qualitatively by a visual analysis. Three volumes of each modality that are not used during training are used to evaluate the process. The four approaches are evaluated and compared to the groundtruth annotations. The evaluation is done separately for CT and MRI scans.
In the three CT scans used, there are a total of 62 annotated lesions. For the three MRI scans, 35 lesions are annotated. Numbers of how many of those lesions can be detected by the models are given.

For the qualitative analysis, the probability maps are overlayed with the slices of the volumes in order to gain an understanding in which regions the networks produce high probability values. Figure 7.3 shows an overlay of a probability map for a single CT slice. The color coding seen in this figure is used throughout the whole analysis. High probability values are depicted in red, low probability values in violet to dark blue.

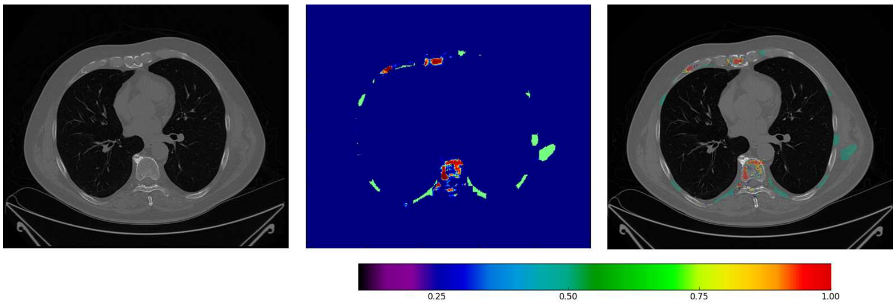


Figure 7.3: *Left:* A CT slice as extracted from the volume. *Center:* The result of the volume parsing. *Right:* Overlayed CT slice.

Hyperparameter	Transfer Learning	Random Initialization
Learning rate (η)	0.0001	0.001
Momentum (α)	0.9	0.9
Drop out rate	0.2	0.2
Minibatch size (m)	64	64
Maximum number of epochs (e)	30	40
Number of frozen Conv Blocks	2	-

Table 7.4: Hyperparameters used during evaluation

7.4 Experiment Setup and Parameter Settings

In this section, the setup of the experiments carried out during evaluation is described. First, the hard- and software used are described. Afterwards, the hyperparameter settings for the networks are discussed.

7.4.1 Hardware and Softwaresetup

All experiments are implemented in *Python3* by using the deep learning library *Keras 2.0* [14] with a *Tensorflow* [2] backend. Keras is chosen as a library because of its user-friendlyness and the possibility to carry out fast experimentations [14]. Hyperparameter search as described in the next section is implemented with the help of *Scikit-learn* [47].

The operating system used is Linux 4.4.0. All networks are trained on a NVIDIA GeForce GTX 960M GPU.

7.4.2 Parameter settings

An overview of the hyperparameters set for the evaluation is given in Table 7.4. The choice of those parameters is described in the following.

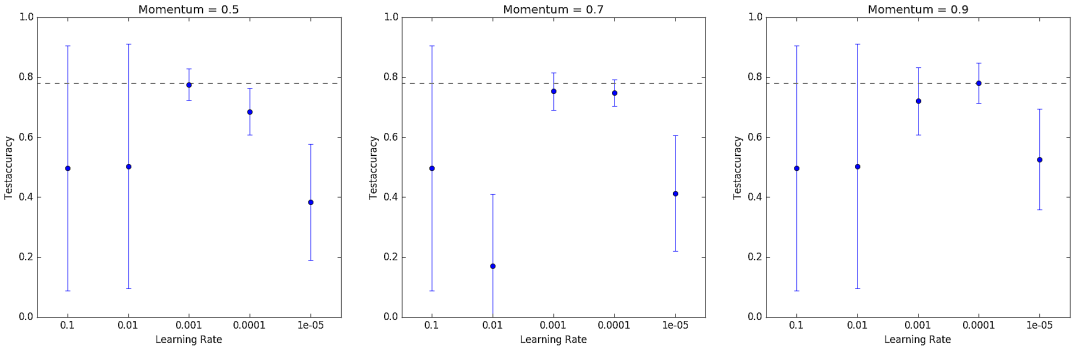


Figure 7.4: Grid search for the best combination of learning rate and momentum. The highest accuracy of 0.78 was found by setting η=0.0001 and α= 0.9

Learning rate and momentum The learning rate and momentum for all models trained with a transfer learning approach is found by a grid search approach. In a preliminary run, combinations of learning rates of η=[0.1, 0.01, 0.001, 0.0001, 0.00001] and momentums of α=[0.5, 0.7, 0.9] are tested. This test run is using the $\mathbf{TL}^{1C}_{CT}$ model and was fixed for all other transfer learning models as well. The results are depicted in Figure 7.4.

Based on the results, the learning rate was fixed to 0.0001 with a momentum of 0.9. The relatively low learning rate when using transfer learning approaches is consistent to literature and is argued by an assumption of good initialization due to the pre-training. A higher learning rate potentially changes this initialization too rapidly and thus the pre-training becomes useless [16].
The learning rate for random initialization was found in a similar approach and was set to 0.001 and with a momentum of 0.9.

Dropout The dropout probability is set to 0.2 for all experiments. This value is found by a testing of multiple values on the $\mathbf{TL}^{1C}_{CT}$ model. The tested values were [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7]. Results for a preliminary experiment with a reduced number of samples and epochs are shown in Figure 7.5. On a limited number of samples the model was trained for 15 epochs and then validated. The accuracy of using a dropout of 0.2 is chosen since it yields the best result.

For comparability the parameter is fixed for all models.

Minibatch size is fixed to 64 for all trained networks.

Number of epochs Since the method used is implemented with early stopping the number of epochs is less critical. Transfer Learning models are trained with a maximum number of 30 fine-tuning epochs. Maximum number of epochs when training from scratch is set to 40.

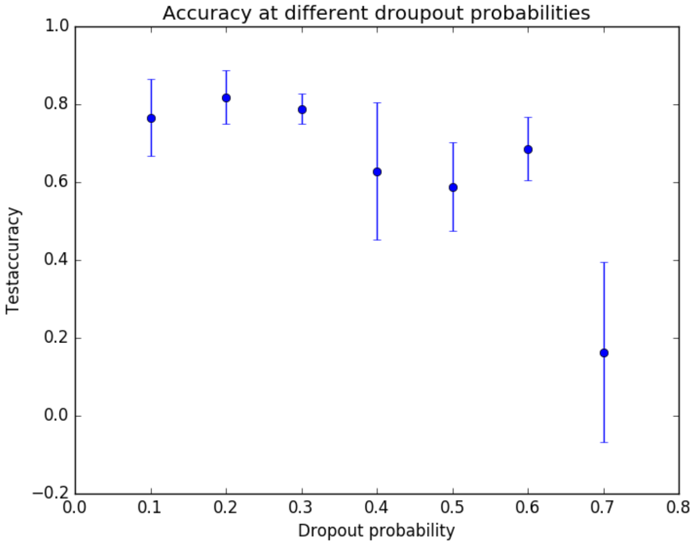


Figure 7.5: Search for the best dropout probability. A value of 0.2 results in the highest accuracy and is chosen as hyperparameter value.

Number of frozen layers For the transfer learning models another critical hyperparameter is the number of layers to freeze during fine-tuning. Since the VGG16 network is organized in blocks of convolutional layers (see Figure 6.4) a preliminary experiment was carried out testing how many of those blocks to freeze. As for the hyperparameters discussed before, the $\mathbf{TL}_{CT}^{1C}$ model was used for the experiment.

Figure 7.6 depicts the result of the experiment. The highest accuracy is reached when freezing two blocks of convolutional layers. This was the final setting for frozen layers in the course of this thesis.

7.5 Summary

In this chapter, the evaluation of the method used in the thesis is described. First, the CT and MRI dataset for training and testing is introduced. It is shown how lesions are annotated and problems with the bone masks provided for MRI volumes are stated. It is important to keep these problems in mind when evaluating the approach. Afterwards, performance measures are introduced which are used to evaluate the method on an image patch level. The evaluation of the volume parsing method is described and how to read the outcome of volume parsing is explained. An important task is how to set the hyperparameters of the network for evaluation, this process is described in Section 7.4.2.

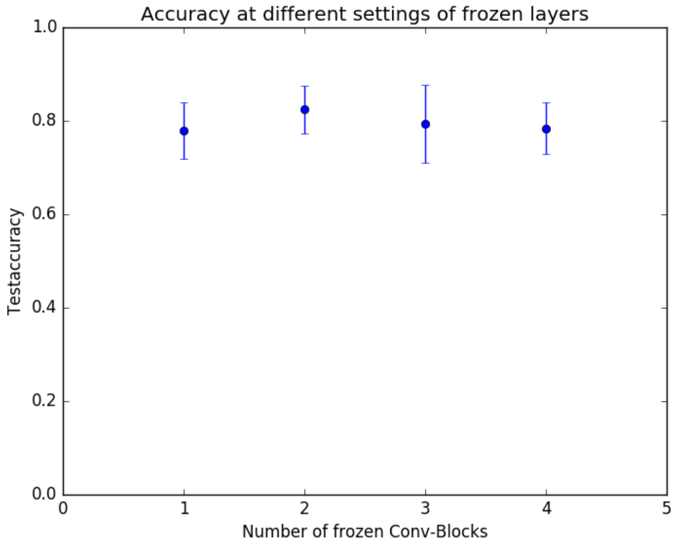


Figure 7.6: Search for a number of frozen layers to freeze during fine-tuning. Freezing two ConvBlocks of the used network reaches the highest accuracy of 0.82.

CHAPTER 8

Results on the VISCERAL Dataset

In this chapter, the results of the evaluation are discussed. Due to the different nature of the image modalities, results on CT volumes and MRI volumes are reported in different sections.
First, results on CT data are discussed, afterwards those for MRI data. For both modalities quantitative results on image patches, as well as a qualitative analysis of the volume parsing method, are given.

8.1 Results on computed tomography data

In this section, results of evaluating the models trained on CT data are given. The results on the image patches given in Section 8.1.1 are computed on the 1068 samples of the test set (see Section 7.1.1). Afterwards, the evaluation of volume parsing of three whole body CT volumes is given.

8.1.1 Results on Patches

The performance measures described in Section 7.2, computed on the test set, are shown in Table 8.1. It is shown that models fine-tuned with transfer learning outperform the corresponding models trained from scratch.

All models are able to detect lesions with high accuracy (>0.83) on image patch level. The three channel approach outperforms the single channel approach. When using the three channel patch extraction, it can be seen that the F-Score and AUC value of using transfer learning and using random initialization are almost equal. F-Score of 0.92 ($\mathbf{TL}_{CT}^{3C}$) and 0.91 ($\mathbf{RI}_{CT}^{3C}$) and AUC of 0.97 for both models. This is an indication that for CT images the three channel patch extraction is a valuable addition to the approach.

	Accuracy	Recall	Precision	F-Score	AUC
$\mathbf{TL}_{CT}^{1C}$	0.89	0.87	0.91	0.89	0.96
$\mathbf{RI}_{CT}^{1C}$	0.83	0.84	0.82	0.83	0.91
$\mathbf{TL}_{CT}^{3C}$	**0.92**	**0.90**	**0.95**	**0.92**	**0.97**
$\mathbf{RI}_{CT}^{3C}$	0.91	**0.90**	0.92	0.91	**0.97**

Table 8.1: Performance measures computed on CT image patches of the test set for all four models.

The $\mathbf{TL}_{CT}^{3C}$ performs best in regard to all performance measures, especially the precision of 0.95 is higher compared to the $\mathbf{RI}_{CT}^{3C}$ method (0.92), which means there are less false negatives when testing with the transfer learning method.

When comparing the ROC Curves in Figure 8.1 it can be seen that the models $\mathbf{TL}_{CT}^{1C}$, $\mathbf{TL}_{CT}^{3C}$ and $\mathbf{RI}_{CT}^{3C}$ clearly outperform the $\mathbf{RI}_{CT}^{1C}$ model trained from scratch on single channel image patches. The three models reach a true positive rate of almost 1 at a false positive rate at around 0.3, where the $\mathbf{RI}_{CT}^{1C}$ model only reaches a TPR of nearly 1 at a high FPR of approximately 0.7. The three channel models clearly outperform the $\mathbf{TL}_{CT}^{1C}$ on a lower false positive rate, which is an indication that the three channel enhancement has more potential than fine-tuning to improve the performance when using CT image patches.

Absolute numbers of the results on the 1068 image patches of the test data set are given in Figure 8.2. All approaches reach high numbers of correct classifications with $\mathbf{TL}_{CT}^{3C}$ having the best result with 982 correct classifications. Only at non-lesion/lesion confusion it misclassifies more non-lesions as lesions than the $\mathbf{RI}_{CT}^{3C}$ model. However, the margin in absolute numbers between the two three channel approaches regarding this kind of confusion is neglectable (59 to 56 misclassifications).

To get an insight of which lesion patches are classified correctly, and which are classified wrongly, some samples are given in the following.
To show the strengths and weaknesses of the CNN approaches, Figure 8.3 shows patches that are classified correctly or misclassified by all trained models. The samples are chosen to capture a wide range of different visual appearances and to show image patches that look similar although they depict positives and negatives. This presents an overview of the dataset used for training and testing.

Multiple observations can be made by looking at the samples in Figure 8.3:
(1) It can be seen that not only strong features, but also subtle variations of patterns are used to correctly distinguish between lesion and non-lesion patches. This becomes especially apparent when comparing *tp-2* and *fp-2*.
(2) The variations of appearance of the lesions is high, when considering the rows *tp* and *fn* of the figure which depict lesions annotated in the groundtruth. Due to this high variation, it is hard to capture the whole range of variance in the limited dataset that

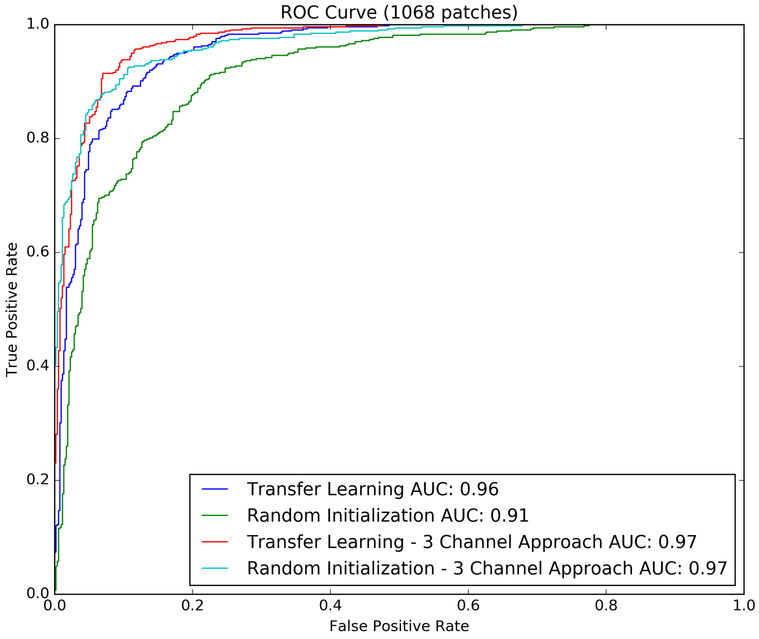


Figure 8.1: ROC curves of all four models computed on CT image patches of the test set.

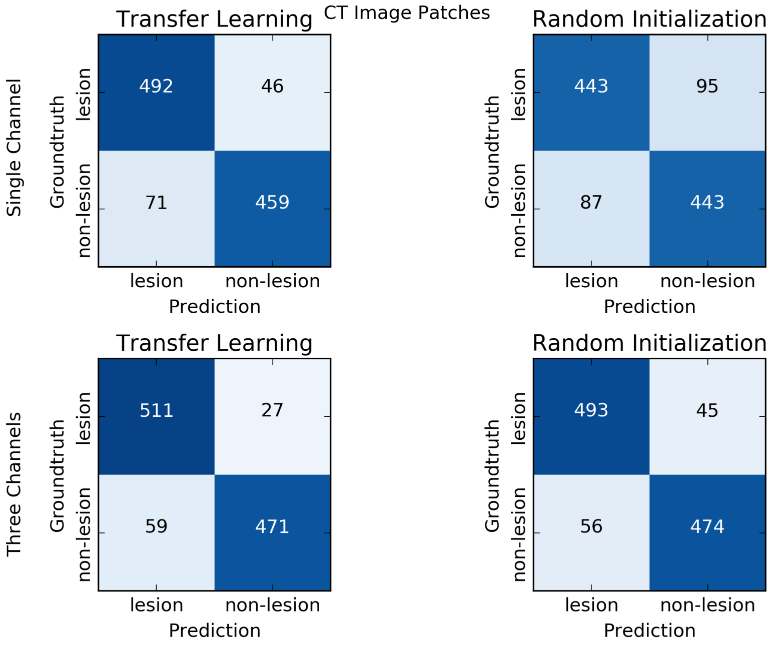


Figure 8.2: Confusion matrices ct image patches

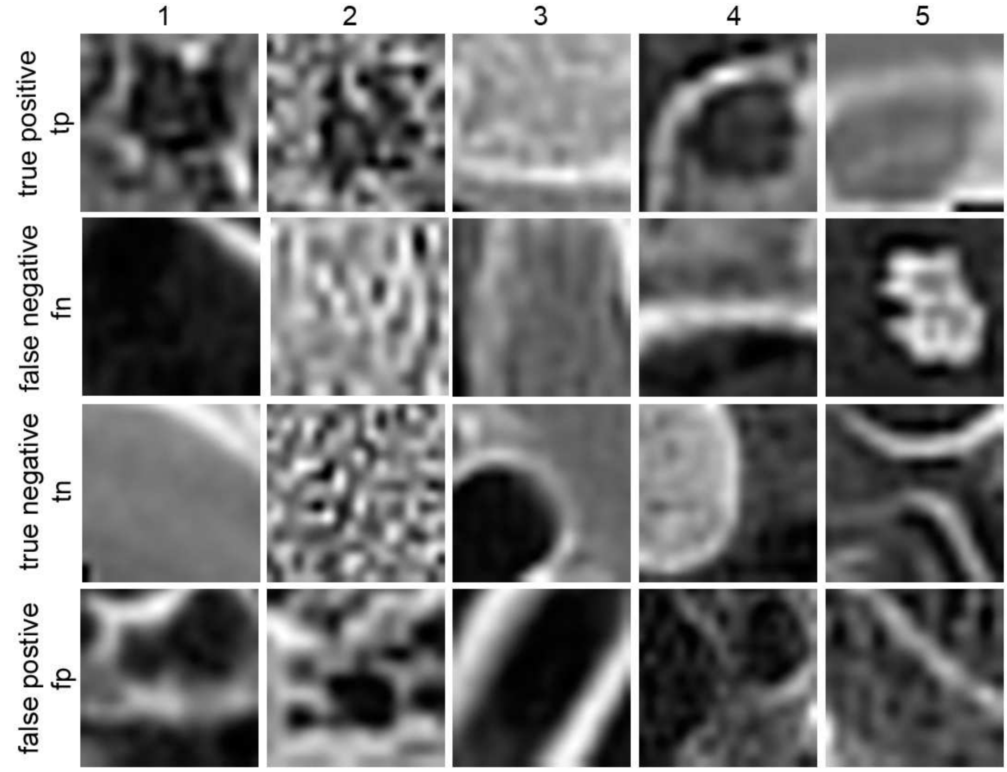


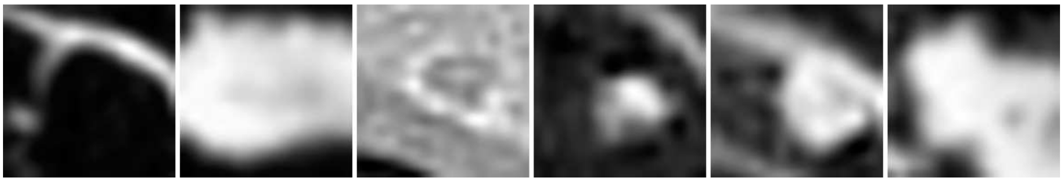
Figure 8.3: CT image patches correct classified or misclassified by all trained models.

is used to train the networks. Variations in appearance of a lesion are due to different stages of progression of a lesion and the disease, and also due to different body regions in which they occur [18]. A limited dataset cannot capture all possible variations in sufficient quantity, some variations are under-represented in the dataset and cannot be learned by the networks.

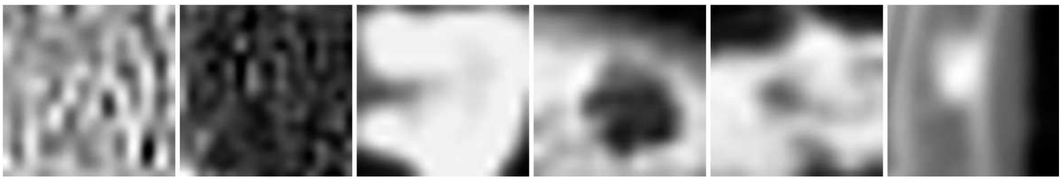
(3) It can be interpreted that the rows of *tp* and *fn* show a similar visual appearance and the confusion of the non-lesion patches of row *fp* is understandable by looking at samples, because visual cues to separate those from the true lesion patches are missing.

(4) The lesion in patch *fn-5* has a clear visual appearance and one might expect that a CNN can classify the lesion correctly. Still, when *fn-5* is compared to the lesions in row *tp* it can be seen that a *fn-5* is brighter than all other lesions. Thus, the misclassification can be explained by the rare visual appearance of *fn-5*. Such an appearance is under represented in the training set, which makes it impossible for the CNN to learn such cases correctly.

(5) Considering column *2* where all patches show a texture-like vision with no clear borders. What is interesting that *tp-2* and *fn-2* which depicts lesions and *tn-2* and *fp-2* (non-lesion patches) have less obvious features in common than the pairs *tp-2*/*fp-2* and *tn-2*/*fn-2*.



(a) Lesion patches that are classified correctly as lesion by the $\mathbf{TL}_{CT}^{1C}$ model, but are misclassified by the $\mathbf{RI}_{CT}^{1C}$ model.



(b) Classified correctly as lesion by $\mathbf{TL}_{CT}^{3C}$ but not $\mathbf{RI}_{CT}^{3C}$.

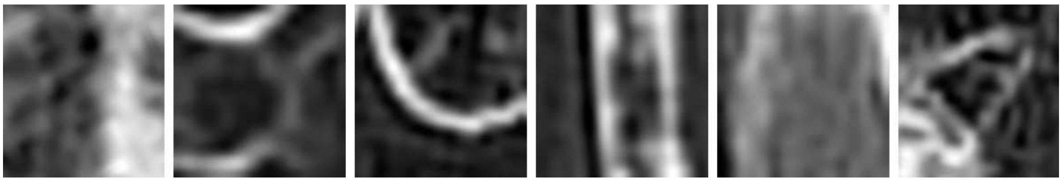
Figure 8.4: Samples for lesions that are correctly identified by transfer learning approaches but not random initialization approaches.

Transfer learning vs. random initialization

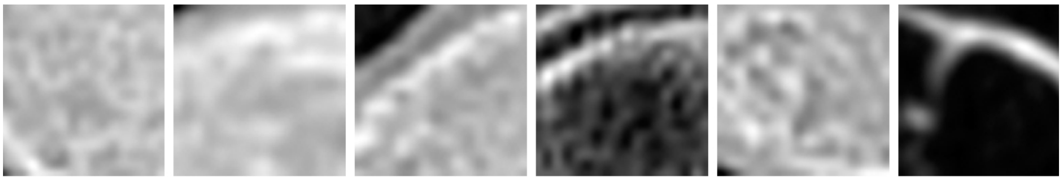
In the following, the effect of transfer learning is analyzed in comparison to random initialization. For that purpose, the results of models with the same patch extraction strategy are compared; those are $\mathbf{TL}_{CT}^{1C}$ and $\mathbf{RI}_{CT}^{1C}$, respectively $\mathbf{TL}_{CT}^{3C}$ and $\mathbf{RI}_{CT}^{3C}$.
For the $\mathbf{TL}_{CT}^{1C}$ model we see in Figure 8.2 that 46 lesions are misclassified as non-lesions in comparison to 95 such misclassifications for $\mathbf{RI}_{CT}^{1C}$. To get an insight into which lesions are misclassified by random initialization approaches see Figure 8.4. Lesions that are classified correctly by the transfer learning approach but not by the random initialization approach are shown. Especially lesions with high HU values and thus brighter appearance are captured more reliably with the transfer learning approach. In general, a comparison of the patches classified as true positives show that the $\mathbf{TL}_{CT}^{1C}$ model is able to capture a wider variety of lesions in terms of their appearance.
A similar observation can be drawn when comparing the results of misclassification of the three channel networks $\mathbf{TL}_{CT}^{3C}$ and $\mathbf{RI}_{CT}^{3C}$ (see Figure 8.4). A wider variety of lesions can be captured by the transfer learning approach.
Random initialization also yields more false positives than transfer learning. In those false positives no clear pattern can be seen.
On image patches, transfer learning outperforms random initialization. There are also samples that are correctly classified as lesions by the random initialization approaches, but not the transfer learning approaches. The quantity of such situations is lower and no clear reasons for the misclassifications can be found.

Single channel patches vs. Three channel patches

In this section, the effect of using three channel patches is discussed. To rule out effects derived by transfer learning, the comparison is done between $\mathbf{TL}_{CT}^{1C}$ and $\mathbf{TL}_{CT}^{3C}$,



(a) Lesion patches that are classified correctly as lesion by the $\mathbf{TL}^{3C}_{CT}$ model, but are misclassified by the $\mathbf{TL}^{1C}_{CT}$ model.



(b) Classified correctly as lesion by $\mathbf{RI}^{3C}_{CT}$ but not $\mathbf{RI}^{1C}_{CT}$.

Figure 8.5: Samples for lesions that are correctly identified by three channel approaches but not single channel approaches.

respectively $\mathbf{RI}^{1C}_{CT}$ and $\mathbf{RI}^{3C}_{CT}$. Both three channel approaches achieve lower numbers of confusions than the corresponding single channel methods; 86 vs. 107 for transfer learning approaches and 101 vs. 182 for random initialization (see Figure 8.2). The reasons for this effect are analysed in the following.

From Figure 8.5 it can be seen that for the random initalization methods lesions which can be detected by changes of texture rather than their clear boundaries are detected more reliably by $\mathbf{RI}^{3C}_{CT}$ than by $\mathbf{RI}^{1C}_{CT}$. This indicates that through the separation of different information in three different channels, an underlying structure to separate those lesions from non-lesions can be found. For the transfer learning methods no such clear indication can be found. There are, however, indications that lesions depicted with low intensity are detected by the single channel approach less reliably than with three channel patches.

8.1.2 Results of Volume parsing

In this section, the results of volume parsing for CT whole body scans are given. As described in Section 7.3, the parsing is evaluated on three whole body scans with a total of 62 lesions annotated.
Table 8.2 shows how many of those lesions are detected by the networks. It is shown that all models can detect lesions with a high recall rate of over 0.85. Still, this comes to the cost of a high number of false positives. Possible reasons for that are discussed in the following qualitative analysis of the visual results.

The results are shown on axial slices, where the CT slice is overlayed with the probability map generated through the volume parsing approach. See Section 7.3 for an explanation how to interpret the results.

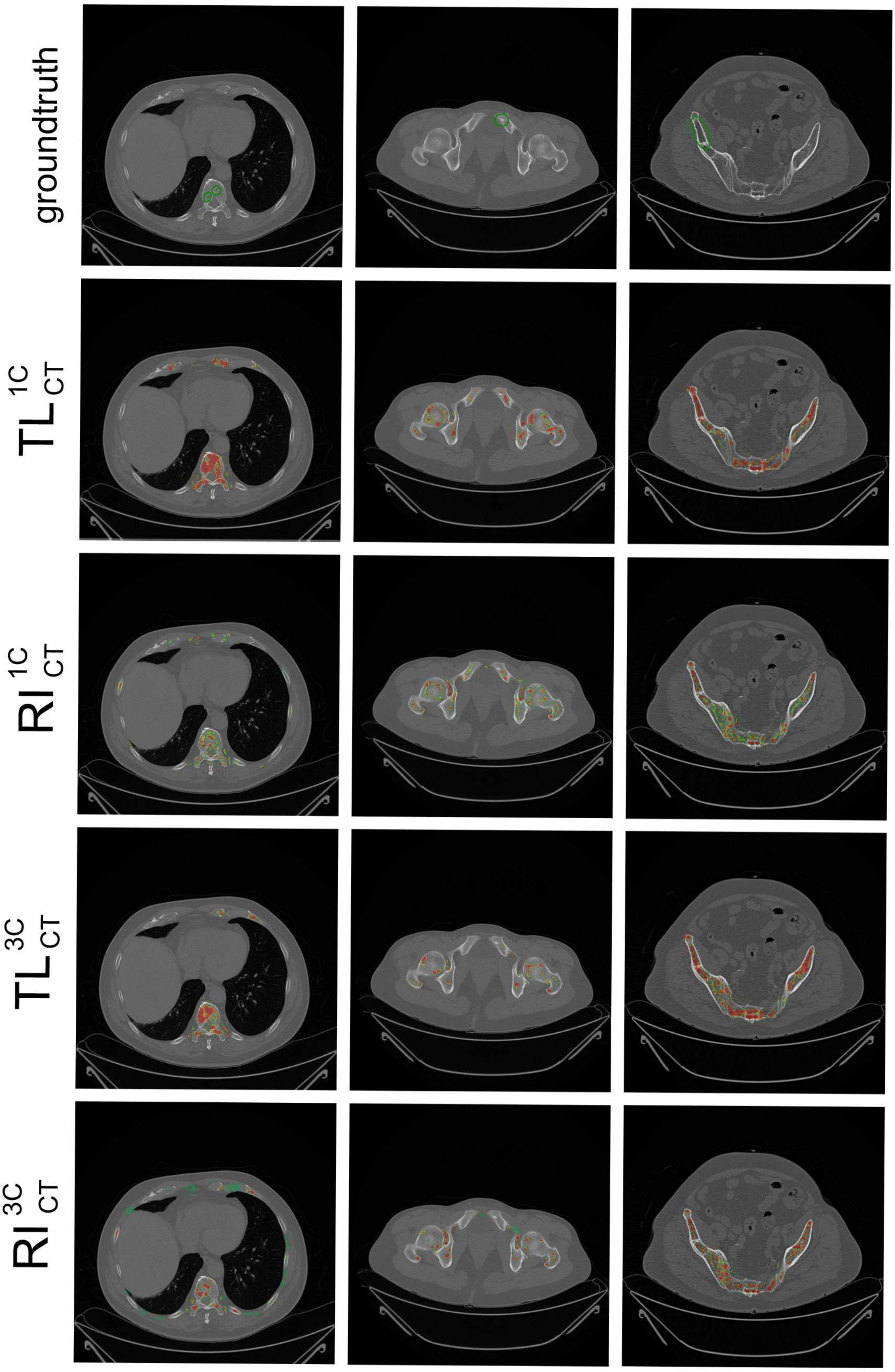


Figure 8.6: Sample result slices of parsing a volume. The comparison shows the groundtruth where the lesions are shown by a green ellipse and the four models that are trained. The second column shows a lesion only detected by $\mathbf{TL}^{1C}_{CT}$.

	Detected	Not-Detected	Recall
$\mathbf{TL}^{1C}_{CT}$	60	2	0.97
$\mathbf{RI}^{1C}_{CT}$	53	9	0.85
$\mathbf{TL}^{3C}_{CT}$	59	3	0.95
$\mathbf{RI}^{3C}_{CT}$	54	8	0.87

Table 8.2: Number of detected lesions in the test volumes.

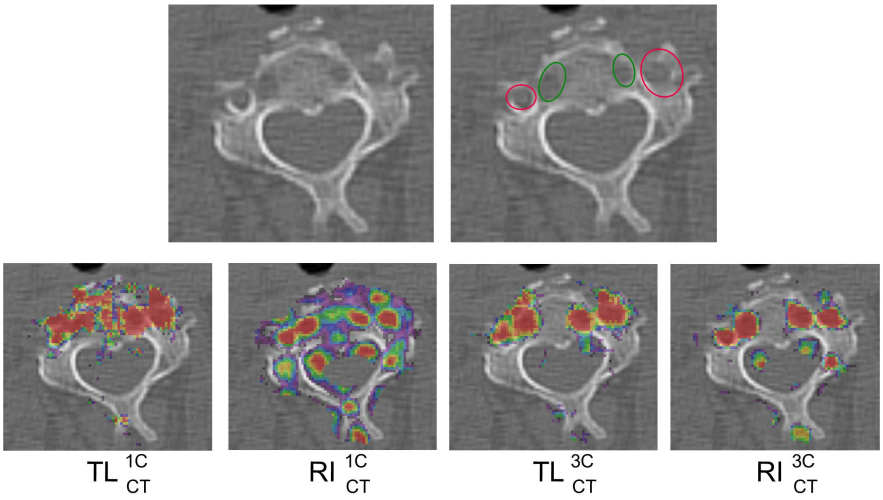


Figure 8.7: Upper row: Region-Of-Interest (ROI) shows a vertebra without annotation and with annotations. Green are lesions. Red are regions that are not annotated as lesions in the groundtruth, but are detected by all models. Lower row: Probability maps of the corresponding regions for the four networks.

Overview in Figure 8.6, the groundtruth of three slices and the results of parsing the slice with the trained models are shown. It can be seen that the $\mathbf{TL}^{1C}_{CT}$, which has the highest recall, produces large regions of high probabilities that are not very specific to real lesion positions. The random initialization approaches produce smaller regions of high probabilities, but a lot of regions with no connection to a lesion in the groundtruth are detected. The $\mathbf{TL}^{3C}_{CT}$ approach also suffers from a high number of false positives, nevertheless, it performs the best detection.
Four outcomes of the visual analysis are discussed in detail in the following: (1) anatomical structures are misclassified as lesions, (2) all models can detect lesions that have a clear visual appearance, (3) there are lesions that are not detectable by any of the trained models and (4) nearby lesions get merged into one region of high probability. In the following, only regions-of-interest instead of whole CT slices are shown to facilitate the comprehension of outcome. Those regions are cut from the slices along the axial plane.

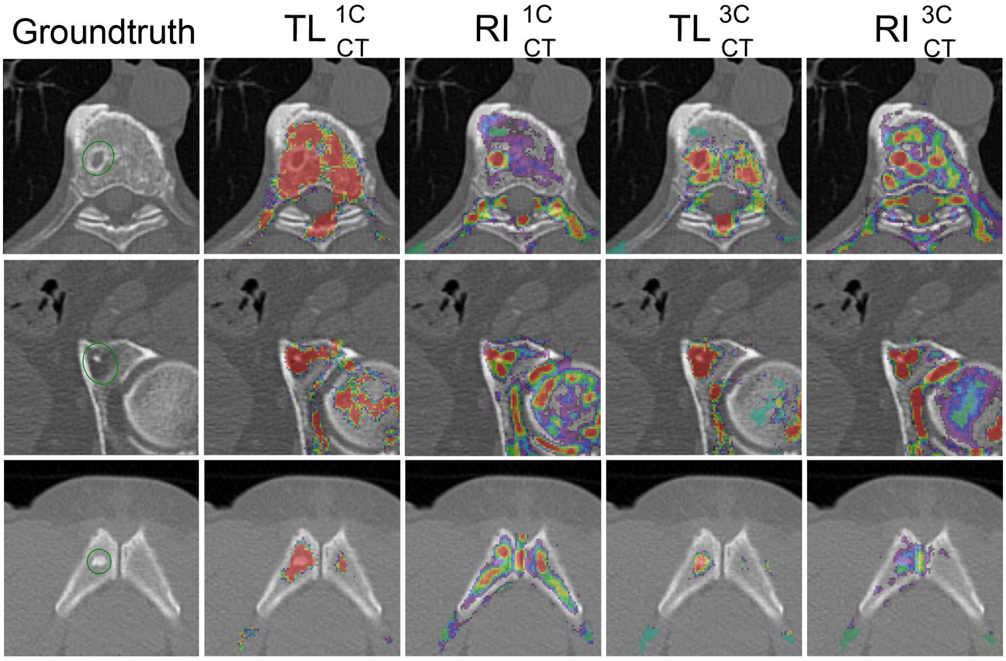


Figure 8.8: Each row shows a lesion annotated in groundtruth data and its detection with all four models. The lesions shown are clearly visible in the CT data, but all have a different visible appearance probably due to the progression of the lesion. $\mathbf{TL}^{3C}_{CT}$ shows the best output on these samples.

(1) Models detect anatomical structures One of the outcomes of the visual analysis is that the models tend to detect anatomical regions that have a visual similarity to lesions. Figure 8.7 illustrates that behaviour. Similar cases can be found multiple times across the volume parsing results. If anatomical regions have a similar appearance to lesions those regions are detected with high probability. All models suffer of the detection of anatomical structures to some extend. This misclassifications can be explained by the limited dataset used. If the dataset only contains few samples to capture such regions, the model cannot adapt to such situations.

(2) Clearly visible lesions Another outcome of the analysis is the detection of clearly visible lesions, as shown in Figure 8.8. The figure shows clear lesions, but they exhibit different appearance. One can see that the $\mathbf{TL}^{1C}_{CT}$ approach has high probability values almost everywhere within the bone, which hints to a bad generalization of the learned model. When looking at all three lesions, $\mathbf{TL}^{3C}_{CT}$ has the best detection results for the lesions annotated. Apparently, the three channel transfer learning approach learns the most variability of lesion appearances. However, also in this model a considerable amount of noise occurs, specially in the first two samples. All models show a lot of false positives. The random initialization networks have a tendency to predict high probability values

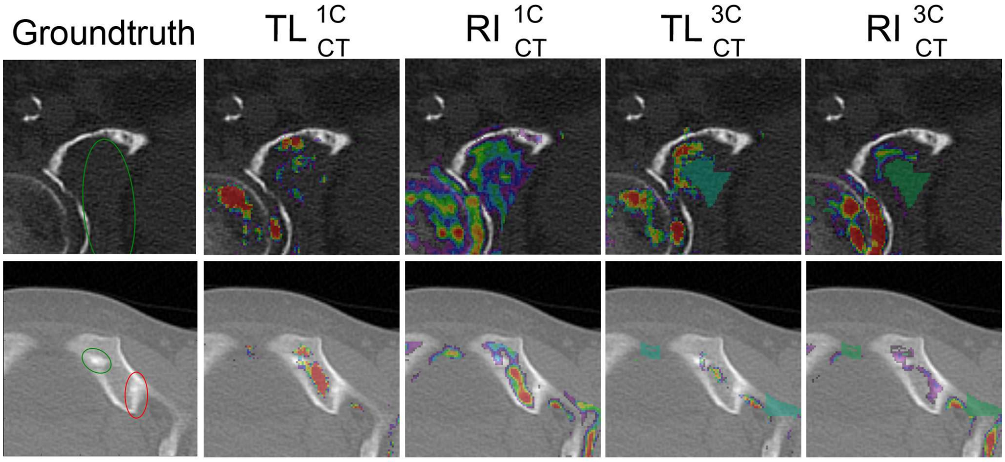


Figure 8.9: Lesions that are not detectable by any of the trained models. *Upper row:* The lesion is large and has an indifferent visual appearance. The receptive field of the volume parsing cannot capture the lesion. *Lower row:* The lesion is small and close to the boundary region. It looks similar to the region circled red in groundtruth which is not a lesion.

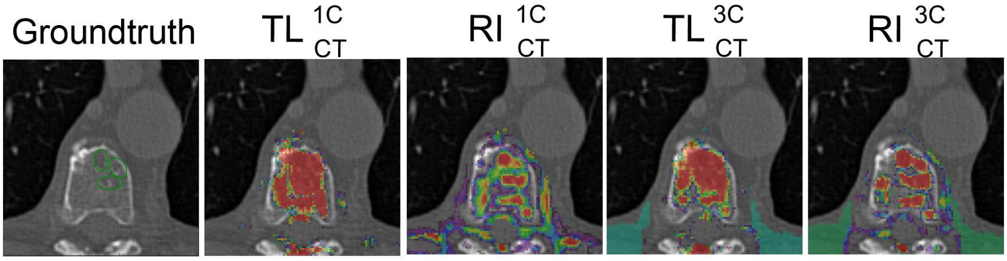


Figure 8.10: Nearby lesions are merged into one region with high probability.

near boundary regions of the bone all across the region of interest.

(3) Lesions not detected by the algorithm Lesions that have a very rare appearance are not detectable by any of the models. Two examples for this situation are shown in Figure 8.9. In the first example, the lesion is atypical in being very massive, so that the receptive field used during volume parsing cannot capture its extent. The second lesion that is not detectable shows a small lesion near the boundary of the bone. The visual appearance is very similar to other boundary regions also shown in the detailed view. Since such appearances are rare, they are seldom represented in the training data and the networks have no chance to learn those cases.

	Accuracy	Recall	Precision	F-Score	AUC
$\mathbf{TL}^{1C}_{MR}$	0.91	**0.93**	0.89	**0.91**	**0.96**
$\mathbf{RI}^{1C}_{MR}$	0.88	0.90	0.83	0.86	0.91
$\mathbf{TL}^{3C}_{MR}$	**0.92**	0.92	**0.91**	**0.91**	**0.96**
$\mathbf{RI}^{3C}_{MR}$	0.89	0.90	0.85	0.87	0.94

Table 8.3: Performance measures computed on MRI image patches of the test set.

(4) Nearby lesions If lesions appear close to each other, no network can separate the lesions. Because of the overlapping field of views during volume parsing, multiple lesions appear as one region of high probability values. One example for a situation of nearby lesions can be seen in Figure 8.10. Since the models are built to detect lesions rather than segment them, the behaviour of the models in such cases is valid.

Summary of volume parsing

It is shown that the volume parsing produces numerous false positives independent on the model used. To determine a model that is superior to the others in case of volume parsing is not possible since each model exhibit strengths and weaknesses depending on the region-of-interest the focus of analysis lies on. One outcome that is consistent over the whole analysis is that $\mathbf{TL}^{1C}_{CT}$ is not suitable since it produces high probability values at numerous positions of the bone that are not connected to a lesion.

8.2 Results on MRI data

In the following section, the results of models trained on image patches extracted from MRI whole body volumes are discussed. First, performance measures on image patches are given. Afterwards, a visual evaluation of three volumes parsed by the detector is given.

8.2.1 Results on Patches

The performance measures computed on MRI image patches are given in Table 8.3. It is shown that models trained with transfer learning outperform those trained from scratch. While the transfer learning approaches perform equally to each other with a F-Score of 0.91 and an AUC of 0.96, the random initialization approach benefits from training on three channel patches in comparison to single channel patches. The recall is nearly equally high for all models, 90% of the lesion patches can be classified correctly. The precision is higher for transfer learning approaches, 0.89 to 0.83 for single channel patches and 0.91 to 0.85 for three channel patches.

The ROC Curves for MRI image patches of the test set are shown in Figure 8.11. As seen in the comparison of performance measures transfer learning approaches outperform

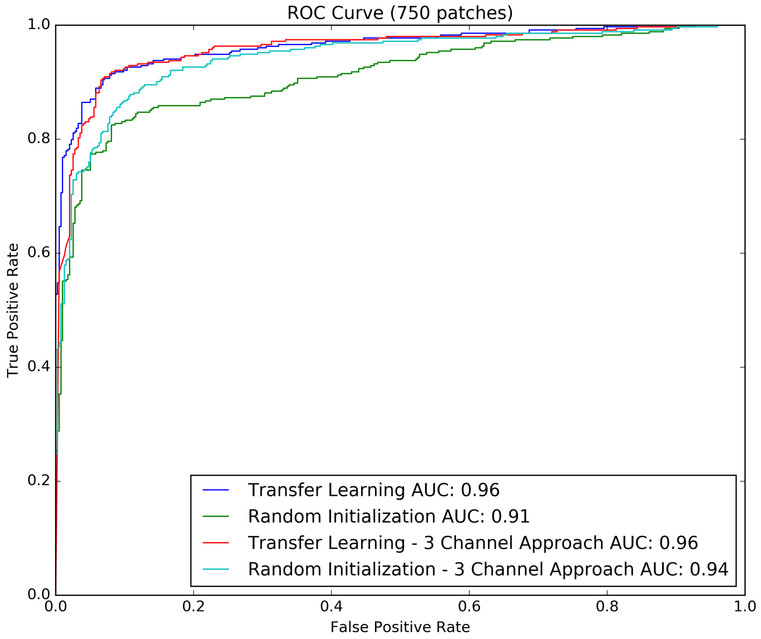


Figure 8.11: ROC Curve computed on a test set of MRT image patches.

random initialization methods. The $\mathbf{TL}_{MR}^{1C}$ and $\mathbf{TL}_{MR}^{3C}$ curves are nearly identical, while the curve of the $\mathbf{RI}_{MR}^{3C}$ model is significantly lower at low false positive rates (less than 0.3). The $\mathbf{RI}_{MR}^{1C}$ performs significantly worse than all other methods at all false positive rates.

A comparison in terms of absolute numbers for the 750 patches of the test set is depicted in Figure 8.12. The comparison shows that transfer learning approaches yield higher numbers of correctly classified samples than random initalization. 686 correct classification by $\mathbf{TL}_{MR}^{1C}$ to 655 by $\mathbf{RI}_{MR}^{1C}$, respectively 689 to 662 on three channel approaches. Also, three channel approaches yield higher numbers than single channel approaches.
To get a deeper insight into what lesion patches look like in the test data set see Figure 8.13. Classifications that are true positives, false positives, false negatives and true negatives in all four network are shown.

From this figure the following observations can be made:
(1) Lesion patches that are misclassified as non-lesions (*fp*) exhibit an appearance where the lesion cannot be detected easily. The variation among the vision of those patches is low, which is a hint that patches with a similar appearance can not be learned correctly by the networks. Possible reasons for that are an under-representation in training data or missing visual cues that separate these patches from true negatives.
(2) In row *fp* patches that cannot be distinguished from *tp* are shown. When comparing

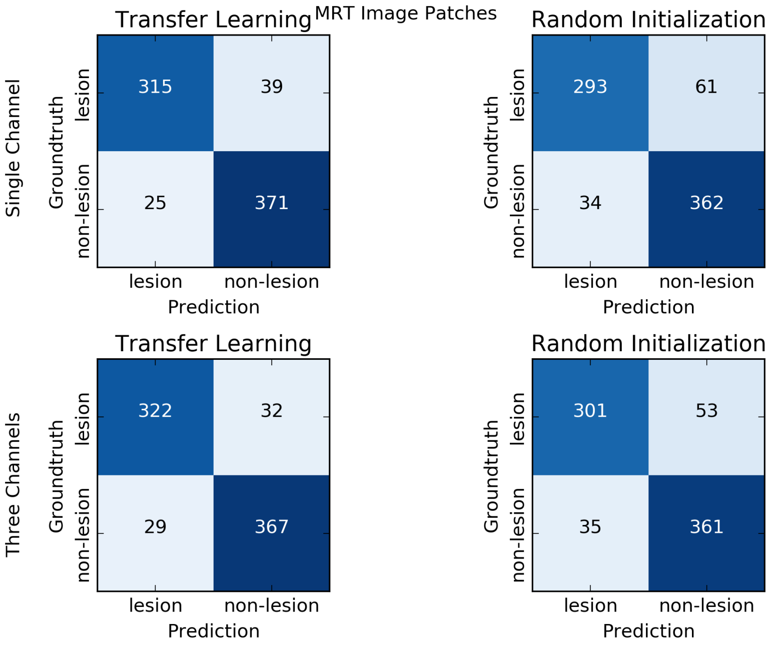


Figure 8.12: Confusion matrices mrt image patches

these two rows, the confusion can be explained by the similar visual appearance.
(3) The sample in *tn-5* is correctly classified as negative by all models although the appearance has similar visual features than the lesions *tp-1*/*tp-2*.

Transfer Learning vs. Random Initialization

To compare the effect of transfer learning to random initialization the results of network $\mathbf{TL}_{MR}^{1C}$ are compared to those of $\mathbf{RI}_{MR}^{1C}$ and those of $\mathbf{TL}_{MR}^{3C}$ to $\mathbf{RI}_{MR}^{3C}$. In terms of performance measures and absolute numbers, transfer learning outperforms random initialization. Some sample cases are shown in the following to get a deeper understanding. Figure 8.14 shows samples from both the three channels and the single channel approaches that were correctly classified as lesions by the transfer learning methods but not the random initialization method.

It can be seen that the transfer learning approach captures lesions with a higher variability. In samples that are correctly classified as non-lesions in transfer learning methods and not in random initialization methods, no pattern can be found and thus no clear reason for the higher number of false positives is found.

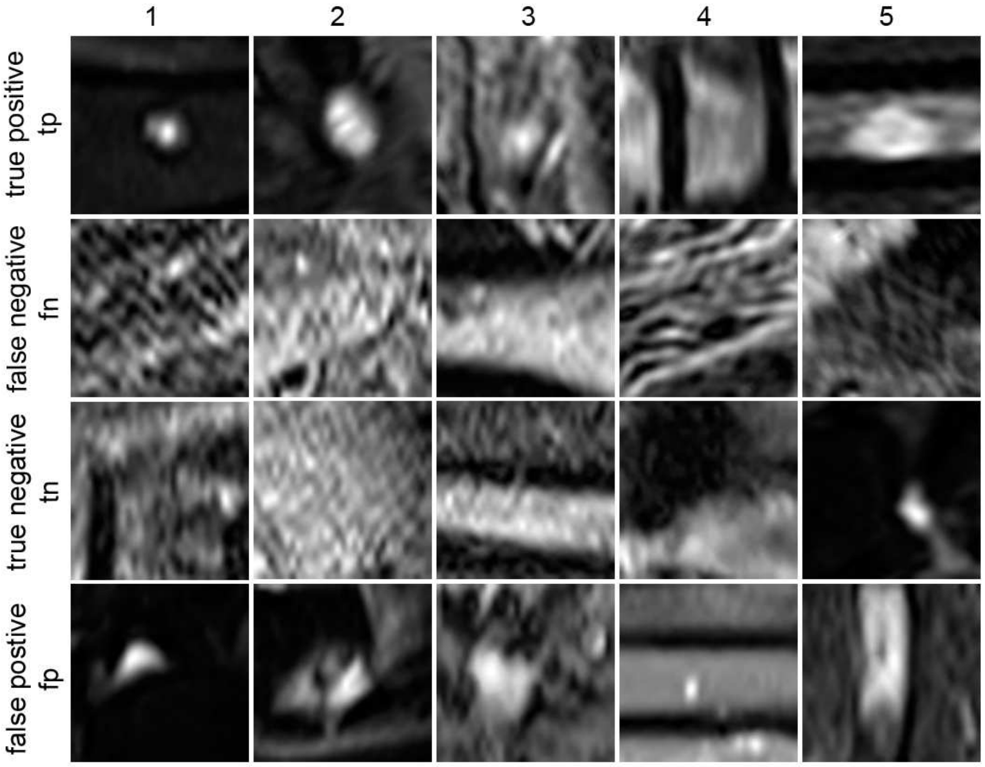
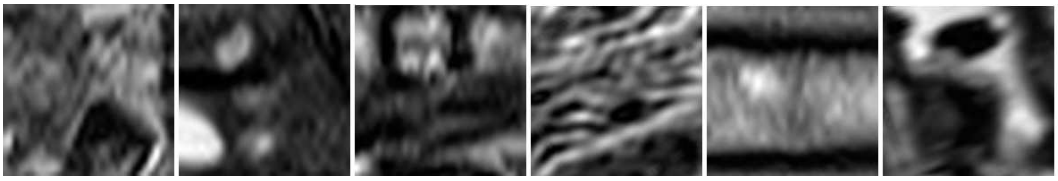
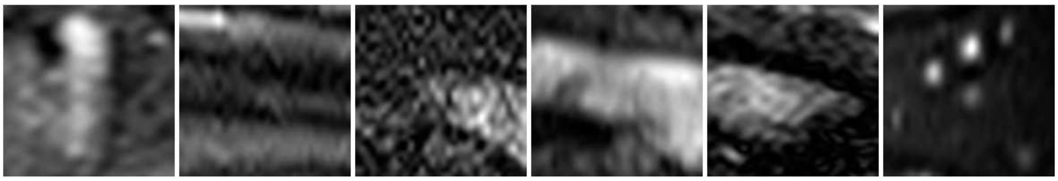


Figure 8.13: Correct classification or misclassification of lesions in MR data by all trained models.

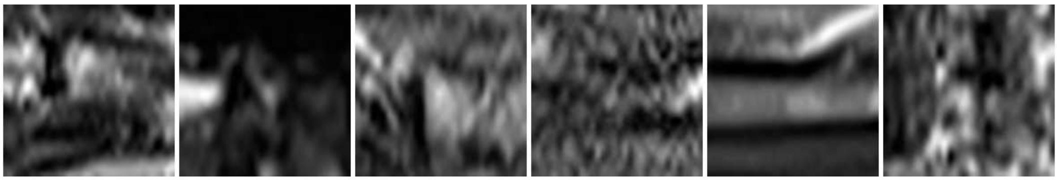


(a) Lesion patches that are classified correctly as lesion by the $\mathbf{TL}^{1C}_{MR}$ model, but are misclassified by the $\mathbf{RI}^{1C}_{MR}$ model.

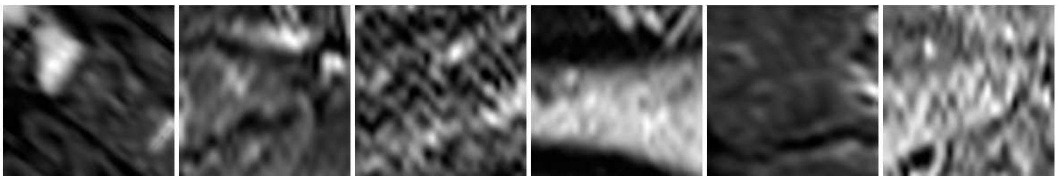


(b) Classified correctly as lesion by $\mathbf{TL}^{3C}_{MR}$ but not $\mathbf{RI}^{3C}_{MR}$.

Figure 8.14: Samples for lesions that are correctly identified by transfer learning approaches but not random initialization approaches.



(a) Lesion patches identified correctly as lesion by the $\mathbf{TL}^{3C}_{MR}$ model, but are misclassified by the $\mathbf{TL}^{1C}_{MR}$ model.



(b) Classified correctly as lesion by $\mathbf{RI}^{3C}_{MR}$ but not $\mathbf{RI}^{1C}_{MR}$.

Figure 8.15: Samples for lesions that are correctly identified by three channel approaches but not single channel approaches.

Single channel patches vs. Three channel patches

To compare the single channel patch approach to the three channel patch approach, models with the same learning protocols are compared ($\mathbf{TL}^{1C}_{MR}$ and $\mathbf{TL}^{3C}_{MR}$ / $\mathbf{RI}^{1C}_{MR}$ and $\mathbf{RI}^{3C}_{MR}$).

In Section 8.15 patches that are classified correctly by three channel approaches, but not the corresponding single channel approach are shown. Almost all samples shown are lesions that can be identified by their texture, rather than their boundary. From this it can be interpreted that the three channel method enables the networks to learn more meaningful features for those lesions, which not rely purely on the shape of the lesion.

8.2.2 Results of volume parsing

In this section, the results of the qualitative analysis of the volume parsing of MRI scans are discussed. In Table 8.4 absolute numbers of detected/non-detected lesions are given. The relatively high number of undetected lesions is mainly due to insufficient bone masks, this issue is discussed further in the following analysis. For that reason, the column *Recall without problematic mask cases* states the recall of lesions that potentially can be detected as they lie inside the bone mask.

Overview Figure 8.16 shows one sample slice of an MRI scan with the groundtruth annotated and the outcome of volume parsing with all models. It can be seen that all approaches are able to detect lesions, but also produce numerous false positives. $\mathbf{TL}^{1C}_{MR}$ shows less noise than other approaches, but the recall of this network is the lowest (0.86).

	Detected	Not-Detected	Recall	Recall without problematic mask cases
$\mathbf{TL}^{1C}_{MR}$	24	11	0.69	0.86
$\mathbf{RI}^{1C}_{MR}$	28	7	0.80	1.0
$\mathbf{TL}^{3C}_{MR}$	26	9	0.74	0.93
$\mathbf{RI}^{3C}_{MR}$	27	8	0.77	0.96

Table 8.4: Number of detected lesions in the MRI test volumes.

From the overview figure it can be seen that the perfect recall (1.0) of $\mathbf{RI}^{1C}_{MR}$ comes with a significant amount of false positives across the slice.

Like in the CT volume parsing analysis (see Section 8.1.2), only regions-of-interest are shown in the following explanations. Main outcomes of the analysis are: (1) although the bone mask is extended, lesions might lie outside the bone, (2) through the extension of the bone mask, regions outside the bone are parsed and the false positive rate is rising, (3) clearly detectable lesions and regions that look like them, (4) spinal discs are detected as lesions, (5) detection along the thigh is challenging and (6) random initialization approaches rely on high intensity values, while transfer learning methods extract more reliable features.

(1) Lesions outside of bone mask As discussed in Section 7.1.2 the bone masks provided for MRI volumes are imprecise. The heuristic of dilating the bone mask temper this problem, but the visual analysis shows a lot of lesions that are still outside the bone mask. Figure 8.17 depicts examples for this problem. Overall, seven of the 35 lesions in the test set lie outside the bone mask after dilation. A further extension of the bone mask could lead to a detection of some of those lesions. However, the detection of regions outside the bone mask, as discussed in the following, advise against a too aggressive extension.

(2) Region outside of bone mask are included through dilation A problem connected to (1) is that regions outside the bone mask get included in the volume parsing through the extension. Regions like parts of intestines or the bladder are included and detected as lesions. Figure 8.18 shows one of those problematic cases. A balance has to be found between (1) and (2). An exact solution to this problem lies outside the scope of the thesis.

(3) Lesions/Non-lesions with clear borders Lesions that are clearly detectable by all approaches are shown in Figure 8.19. The lesions are easy to detect and clearly separated from their neighbourhood. Still, there are also regions where no lesions occur that show a similar visual. Those regions are marked as lesions by all methods. A separation of those patches is not possible only based on their visual cues, thus expert knowledge is needed to classify such lesions reliably.

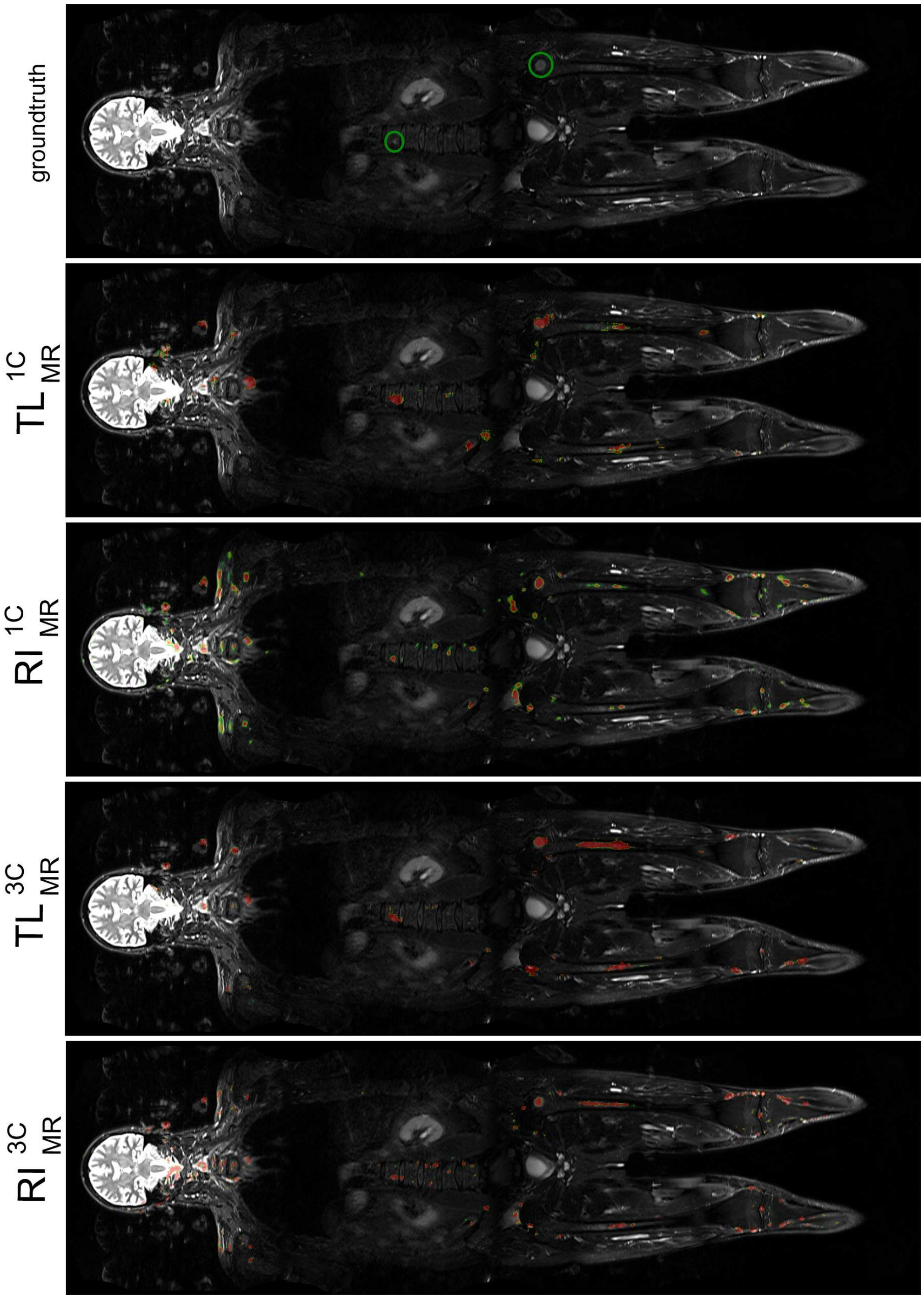


Figure 8.16: Sample outcome slice of parsing a whole volume. The groundtruth is shown in comparison to the outcome of the four models trained. Annotated lesions in the groundtruth are depicted with red circles.

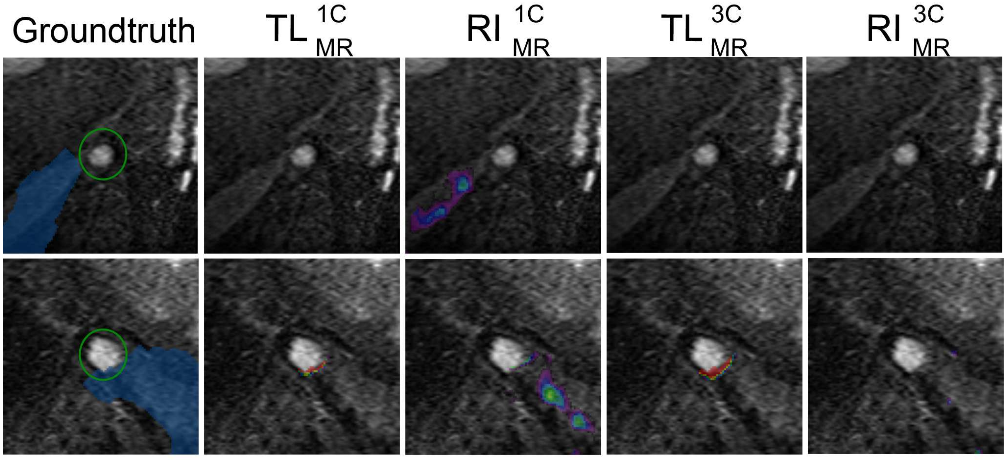


Figure 8.17: In the groundtruth, the dilated bone mask is shown in blue; the lesion is circled in green. Even when dilation is done, the lesion lies outside of the mask. *Second row:* The potential for successful lesion detection can be seen in the transfer learning methods: On the boundary of the mask the network generates high probability values.

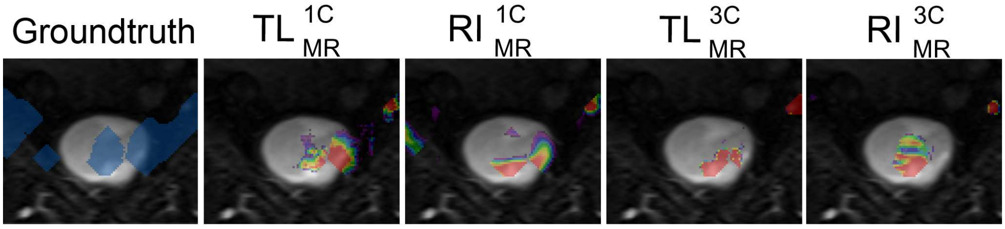


Figure 8.18: The bladder is detected as lesion because of the extension of the bone mask and its similar texture.

(4) Spinal disc detected as lesions An area where lot of misclassification occurs is the spinal column. As seen in Figure 8.20 the spinal discs are often confused with lesions. Especially the random initialization approaches confuse spinal discs frequently, while transfer learning methods are more prone to such misclassifications. This hints to the extraction of more meaningful features by the transfer learning approaches.

(5) Detection along the thigh A second anatomical region in which the approaches exhibit different behaviour is the thigh. Figure 8.21 shows different regions-of-interest extracted along the thigh. It is depicted that the $\mathbf{RI}^{1C}_{MR}$ model has a good detection of the lesion, while the other approaches show a lot of noise. Especially the three channel transfer learning fails to produce a meaningful detection, as high probability values are detected along the whole bone. Figure 8.21 shows just one sample of this behaviour that can be found throughout the three test volumes.

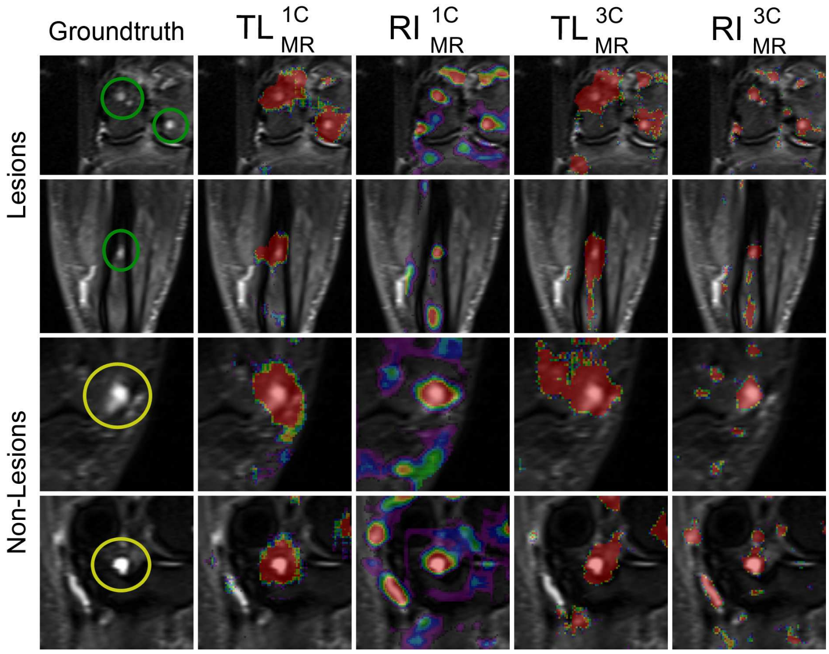


Figure 8.19: *Two upper rows:* True lesions with a clear boundary. *Lower two rows:* Non-lesions with a similar appearance, circled in yellow in the groundtruth

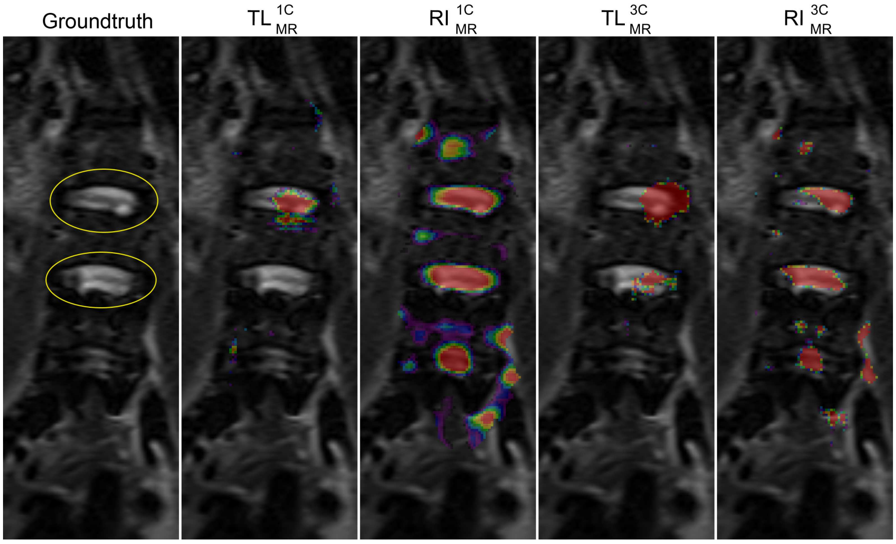


Figure 8.20: Confusion of the spinal disc with lesions. In the groundtruth the spinal discs are circled in yellow. The random initialization approaches strongly respond to spinal discs. The transfer learning approaches show a respond, but are more prone to misclassification of spinal discs.

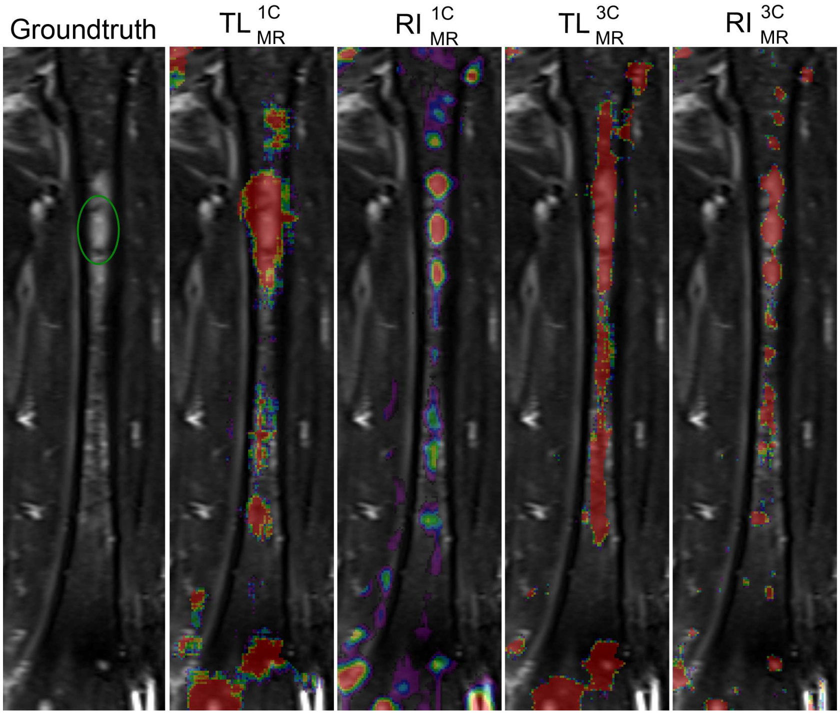


Figure 8.21: Lesions in the thigh are detected by all approaches, but numerous of wrong high probabilities are detected by all approaches expect the $\mathbf{RI}^{1C}_{MR}$.

(6) Random Initialization relies on high intensity In general, the visual analysis shows that random initialization approaches rely on high intensity for the detection of lesions. As long as the intensity is high, the probability value generated by $\mathbf{RI}^{1C}_{MR}$ and $\mathbf{RI}^{3C}_{MR}$ is high. Transfer learning approaches learn more meaningful features taking not only the intensity into account, but also e.g. shape. Figure 8.22 show regions of such confusions by the random initialization methods. The samples are drawn from different anatomical regions to show that this phenomenon is not restricted to certain body parts.

Summary of volume parsing

Important outcomes of the analysis of volume parsing are:

1. No single model outperforms in every case, all models have different strengths and weaknesses depending on the individual visual appearance of the lesions.
2. Randomly initialized models rely too strongly on high intensity values as a feature to distinguish between lesion and non-lesion.
3. Without solving the issues related to the bone mask the analysis is limited.

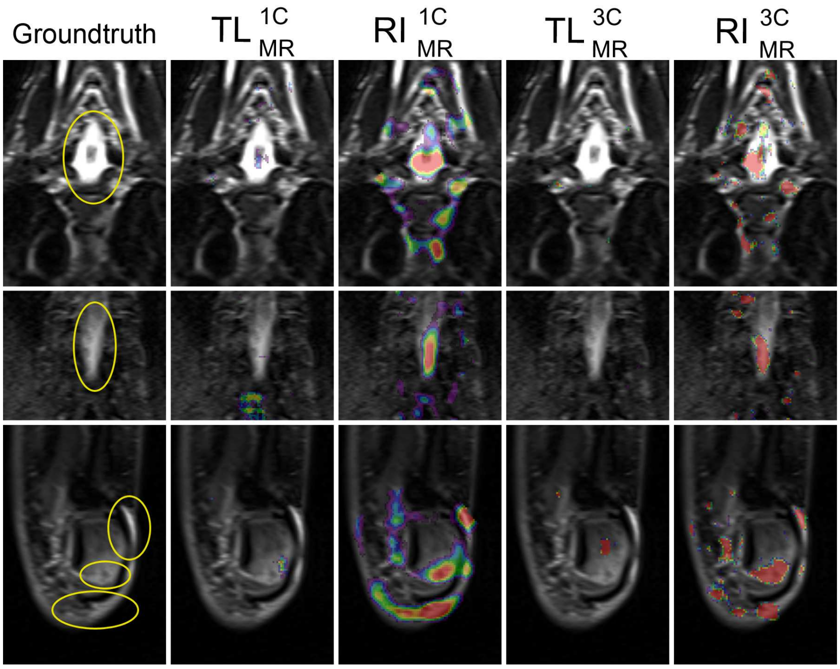


Figure 8.22: The random initialization approaches strongly rely on high intensity values to produce high probability output values. This phenomenon is not restricted to a certain body region. as seen in the samples. *First row:* Neck region. *Second row:* Along the spinal cord. *Third row:* In the knee.

8.3 Summary

In this chapter, the results of the evaluation on CT data and MRI data are shown. A quantitative analysis on image patches shows, that transfer learning can improve the performance of the method. Also, the three channel patch extraction is successfully used to enhance the detection of lesions. The qualitative analysis show the weaknesses of the method, which is the high false positive rate independent on which model is used for detection.

CHAPTER 9

Summary and Discussion

In this chapter, a summary and discussion of the main results of this thesis is given. Also, the research questions stated in Section 1.2 are discussed. The ability to detect lesions is described. Transfer learning is compared to random initialization, as well as a single channel patch extraction to a three channel patch extraction. These comparisons summarize results of both modalities. Afterwards, limitations of the method are described.

9.1 Ability to detect lesions

Is it possible to detect bone lesions in CT and MRI data with the help of convolutional neural networks?

From the results seen in Chapter 8, this question can be answered with yes, it is possible to detect bone lesions with CNNs. Although all shown models have weaknesses in detecting some lesions, the overall results are promising.
All models reach a high recall on image patches ranging from 0.84-0.93. Further improving that results would require more training data to adapt to lesions which have a rare appearance. This high recall rates are also manifested in the volume parsing evaluation. However, during volume parsing the number of false positives is too high to use the method without further improvements. Especially models with high detection rate on whole body volumes also produce a high FPR. To solve that, a way of decreasing the FPR without loosing true positives is needed. The highest F-Scores reached are around 0.90, going significantly higher seems to be impossible with the methods and the limited dataset described in this thesis.

9.2 Transfer Learning vs. Random Initialization

Does transfer learning help to improve the performance of bone lesion detection with CNNs, even though the network is pre-trained on natural images?

The results of the thesis show, that each transfer learning approach outperforms the corresponding random initialization approach. The margin between the F-Score of the transfer learning approaches to the corresponding random initialization evaluated on image patches lies between 0.01 and 0.06. These results are consistent to effects of fine-tuning shown in literature [61] [15]. A positive effect of transfer learning can be observed. As shown in the qualitative analysis, one of the reasons for that lies in the wider variability of lesion appearances learned by transfer learning approaches.
When compared on volume parsing both transfer learning networks detect almost all lesions in CT data, but they also produce more noise than the random initialization results. Nevertheless, a noise reduction strategy is needed on all volume parsing approaches, so it is preferable to use the transfer learning networks.
On MRI volumes the numbers of detected lesions of volume parsing would advise to use a random initialization approach. However, a closer analysis reveals that transfer learning methods learn more meaningful features of lesions while the random initialization rely strongly on high intensity values. Also, the evaluation of volume parsing for MRI is influenced by the imprecise bone mask provided.

Overall the answer to the research question is: Yes, transfer learning improves the performance. This is due the wider variety of lesions captured and due to the fact that more meaningful features are learned by transfer learning approaches, compared to their corresponding random initialization methods.

9.3 Single Channel Image Patches vs. Three Channel Image Patches

In this section, the use of a patch extraction strategy that focuses on different information in each input channel of the network is discussed.

There is indication in the results, that using three channel patches is superior over the use of single channel patches. Especially on CT image patches a positive effect of three channel patches can be observed. Regarding the performance measures computed on image patches, all three channel approaches outperform their corresponding single channel approaches. The main reason for these results is the detection of lesions identified by their texture. Three channel patches are more suitable for detecting lesions that are not characterized solely by the shape boundary but their texture. This effect can be seen on MRI and as well as on CT image patches. In a comparison of the results of parsing whole body volumes, this effect is not observable that clearly. Over all volumes parsed, there is a tendency that three channel networks produce less false positives than their corresponding single channel approaches. However, this is very individual on what region-of-interest the focus lies on.

The main result of the comparison is that, it is valuable to enhance the image patches by three channel extraction. Nevertheless, the observability of the positive effects has a strong variability.

9.4 Limitations

All methods used and the evaluation underlie certain limitations which are discussed in this section. Two limitations regarding the dataset are the size of the dataset and the imprecise bone mask in MRI data. The relatively small dataset is problematic especially for the extraction of positive samples. Only 660 lesions are annotated in CT data and 363 lesions in MRI data. Negative samples could be drawn without limitations. However, a dataset that is imbalanced can lead to problems during training. The imprecise bone mask that is provided for MRI scans prevents lesions to get detected during volume parsing.

The need of a bone mask is one of the limitations of the approach itself. The method focuses on bone regions and cannot be used to detect lesions in whole body images without a preceding segmentation of the bones.

Another limitation shown in the evaluation is that the detection in volume parsing is very individual and depends on the lesion. No model is preferable in all situations.

9.5 Summary

In this section, the research question are answered. The ability of detecting lesions in general is discussed. Afterwards it is summarized how transfer learning and the three channel patch extraction improve the results of detection. The limitations described show that the method needs further improvements to be reliably used to detect lesions.

If one approach should be recommended over the others that would be the three channel transfer learning method. Both strategies transfer learning and three channel extraction has proven as valuable.

CHAPTER 10

Conclusion and Future Work

In this thesis a method to detect bone lesions in Multiple Myeloma patients is introduced. A deep learning approach using convolutional neural networks is described and evaluated. The method utilizes transfer learning to improve the performance of the network. In the first part of the thesis, state-of-the-art methods used in the course of the work are described. First, basics of machine learning in general are discussed. Afterwards, feedforward neural networks and their training are described in detail in order to comprehend convolutional neural networks. CNNs and their main concepts described. These main concepts, sparse connectivity, parameter sharing, pooling and the use of a deep architecture, are discussed in detail. Also, hyperparameters that influence the behaviour and capacity of convolutional networks are introduced. Next, the concept of transfer learning is described and how to use this learning protocol of CNNs is discussed in detail.

The preliminary experiments documented show that fine-tuning is the most suitable transfer learning strategy for detecting bone lesions in medical imaging data. Another experiment shows that the VGG-16 network has the highest potential to be adapted to the task of bone lesion detection.

The methodology used in the thesis is described. Two different patch extraction strategies are used: Single channel patch extraction and three channel patch extraction. To evaluate the improvements of the performance two learning protocols, learning from scratch and fine-tuning are utilized. For each modality a set of four models is trained and used in the volume parsing method described.

Afterwards, the data used for evaluation is described. Evaluation is done quantitatively on image patches and qualitatively on whole body scans that are parsed. The setup and the parameter settings used for the evaluation is described in detail. Next, results for all trained models are shown and compared. The quantitative results show the positive effects of transfer learning and the three channel patch extraction. The qualitative results of the volume parsing approach shows strength and weaknesses of the models.

Finally, it is discussed that the method is suitable to detect lesions and that both, transfer learning and the three channel patch extraction, are valuable additions to the method. Also limitations on the dataset and the method itself are discussed.

10.1 Future Work

In this section the directions of possible future work building on this thesis are discussed.

Dataset: If possible, the dataset should be enhanced and extended. Especially the bone mask for the MRI data has to be improved.

Patch Extraction: A possible improvement in context of patch extraction is to focus on extracting negative patches that are more likely confused as positives. As seen in this thesis, these patches lie in boundary regions of the bone and in regions of anatomical structures that have similar vision to lesions.
To keep the dataset in balance a possible approach would be to extract negative patches in different categories, e.g. boundary, bone marrow or spinal disc, and try to learn a multi-class problem. A similar approach has shown the potential to improve performance in [25].

Post-processing after volume parsing: A post processing method to filter noise and improve the results can be another direction of future work.

Bibliography

[1] Criteria for the classification of monoclonal gammopathies, multiple myeloma and related disorders: a report of the International Myeloma Working Group. *British Journal of Haematology*, 121(5):749–757, 2003.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] Muhammad Jamal Afridi, Arun Ross, and Erik M. Shapiro. On automated source selection for transfer learning in convolutional neural networks. *Pattern Recognition*, 73:65–75, 2018.

[4] Joseph Antony, Kevin McGuinness, Noel E O'Connor, and Kieran Moran. Quantifying radiographic knee osteoarthritis severity using deep convolutional neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 1195–1200. IEEE, 2016.

[5] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2654–2662. Curran Associates, Inc., 2014.

[6] Yaniv Bar, Idit Diamant, Lior Wolf, and Hayit Greenspan. Deep learning with non-medical training used for chest pathology identification. In *Medical Imaging 2015: Computer-Aided Diagnosis*, volume 9414. International Society for Optics and Photonics, 2015.

[7] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7700 LECTURE NO:437–478, 2012.

[8] Christopher M Bishop. *Pattern Recognition and Machine Learning*, volume 4. 2006.

[9] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[10] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[11] Natalie S. Callander and G. David Roodman. Myeloma bone disease. *Seminars in hematology*, 38(3):276–85, 2001.

[12] Gustavo Carneiro, Jacinto Nascimento, and Andrew P Bradley. Unregistered Multiview Mammogram Analysis with Pre-trained Deep Learning Models. In Nassir Navab, Joachim Hornegger, William M Wells, and Alejandro F Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III*, pages 652–660. Springer International Publishing, Cham, 2015.

[13] Rich Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998.

[14] François Chollet et al. Keras. `https://github.com/keras-team/keras`, 2015.

[15] Hiba Chougrad, Hamid Zouaki, and Omar Alheyane. Deep Convolutional Neural Networks for breast cancer screening. *Computer Methods and Programs in Biomedicine*, 157:19–30, 2018.

[16] Brian Chu, Vashisht Madhavan, Oscar Beijbom, Judy Hoffman, and Trevor Darrell. Best practices for fine-tuning visual classifiers to new domains. In *European Conference on Computer Vision*, pages 435–442. Springer, 2016.

[17] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[18] Meletios Dimopoulos, Evangelos Terpos, Raymond L. Comenzo, Patrizia Tosi, Meral Beksac, Orhan Sezer, David Siegel, Henk Lokhorst, Shaji Kumar, S. Vincent Rajkumar, Ruben Niesvizky, Lia Aangela Moulopoulos, and Brian G.M. Durie. International myeloma working group consensus statement and guidelines regarding the current role of imaging techniques in the diagnosis and monitoring of multiple Myeloma. *Leukemia*, 23(9):1545–1556, 2009.

[19] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[20] Qi Dou, Hao Chen, Lequan Yu, Lei Zhao, Jing Qin, Defeng Wang, Vincent C.T. Mok, Lin Shi, and Pheng Ann Heng. Automatic Detection of Cerebral Microbleeds from MR Images via 3D Convolutional Neural Networks. *IEEE Transactions on Medical Imaging*, 35(5):1182–1195, 2016.

[21] Julie C. Dutoit and Koenraad L. Verstraete. MRI in multiple myeloma: a pictorial review of diagnostic and post-treatment findings. *Insights into Imaging*, 7(4):553–569, 2016.

[22] Leila H. Eadie, Paul Taylor, and Adam P. Gibson. A systematic review of computer-assisted diagnosis in diagnostic cancer imaging. *European Journal of Radiology*, 81(1):70–76, 2012.

[23] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[24] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[25] Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Modeling the intra-class variability for liver lesion detection using a multi-class patch-based CNN. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10530 LNCS:129–137, 2017.

[26] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feed-forward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[27] Aaron Goodfellow, Ian, Bengio, Yoshua, Courville. Deep Learning. *MIT Press*, 2016.

[28] Varun Gulshan, Lily Peng, Marc Coram, Martin C. Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, Ramasamy Kim, Rajiv Raman, Philip C. Nelson, Jessica L. Mega, and Dale R. Webster. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA - Journal of the American Medical Association*, 316(22):2402–2410, 2016.

[29] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[30] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[31] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[32] Edward Kim, Miguel Corte-Real, and Zubair Baloch. A deep semantic mobile application for thyroid cytopathology. In *Medical Imaging 2016: PACS and Imaging Informatics: Next Generation and Innovations*, volume 9789, page 97890A. International Society for Optics and Photonics, 2016.

[33] Markus Krenn, Katharina Grünberg, Oscar Jimenez-del Toro, András Jakab, Tomàs Salas Fernandez, Marianne Winterstein, Marc-André Weber, and Georg Langs. *Datasets Created in VISCERAL*, pages 69–84. Springer International Publishing, Cham, 2017.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F Pereira, C J C Burges, L Bottou, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[35] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[36] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[37] Yann LeCun, Ofer Matan, B Boser, John S Denker, D Henderson, RE Howard, W Hubbard, LD Jacket, and HS Baird. Handwritten zip code recognition with multilayer networks. In *Proceedings of the 10th International Conference on Pattern Recognition*, volume 2, pages 35–40. IEEE, 1990.

[38] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42(December 2012):60–88, 2017.

[39] Shih-Chung B Lo, Shyh-Liang a Lou, Jyh-Shyan Lin, Matthew T Freedman, Minze V Chien, and Seong K Mun. Artificial convolution neural network techniques and applications for lung nodule detection. *IEEE Transactions on Medical Imaging*, 14(4):711–718, 1995.

[40] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[41] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[42] Afonso Menegola, Michel Fornaciali, Ramon Pires, Flávia Vasques Bittencourt, Sandra Avila, and Eduardo Valle. Knowledge transfer for melanoma screening with deep learning. In *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, pages 297–300. IEEE, 2017.

[43] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571. IEEE, 2016.

[44] Tom M Mitchell. *Machine learning.* McGraw-Hill, New York, 1997.

[45] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1717–1724, 2014.

[46] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[47] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathiue Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[48] Matthew J. Pianko, Evangelos Terpos, David G. Roodman, Chaitanya R. Divgi, Sonja Zweegman, Jens Hillengass, and Suzanne Lentzsch. Whole-body low-dose computed tomography and advanced imaging techniques for multiple myeloma bone disease. *Clinical Cancer Research*, 20(23):5888–5897, 2014.

[49] Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[50] Lorien Y Pratt, Jack Mostow, Candace A Kamm, and Ace A Kamm. Direct transfer of learned information among neural networks. In *AAAI*, volume 91, pages 584–589, 1991.

[51] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 512–519, 2014.

[52] Adria Romero-Lopez, Xavier Giro-i Nieto, Jack Burdick, and Oge Marques. Skin Lesion Classification from Dermoscopic Images Using Deep Learning Techniques. *Proceedings of the lASTED International Conference Biomedical Engineering (BioMed 2017) February 20 - 21, 2017*, pages 49–54, 2017.

[53] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[54] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408, 1958.

[55] Holger R. Roth, Le Lu, Jiamin Liu, Jianhua Yao, Ari Seff, Kevin Cherry, Lauren Kim, and Ronald M. Summers. Improving Computer-Aided Detection Using Convolutional Neural Networks and Random View Aggregation. *IEEE Transactions on Medical Imaging*, 35(5):1170–1181, 2016.

[56] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.

[57] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference on Information Processing in Medical Imaging*, pages 146–157. Springer, 2017.

[58] Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Geert Litjens, Paul Gerke, Colin Jacobs, Sarah J. van Riel, Mathilde Marie Winkler Wille, Matiullah Naqibullah, Clara I. Sanchez, and Bram van Ginneken. Pulmonary Nodule Detection in CT Images: False Positive Reduction Using Multi-View Convolutional Networks. *IEEE Transactions on Medical Imaging*, 35(5):1160–1169, 2016.

[59] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[60] Wei Shen, Mu Zhou, Feng Yang, Caiyun Yang, and Jie Tian. Multi-scale convolutional neural networks for lung nodule classification. In *International Conference on Information Processing in Medical Imaging*, pages 588–599. Springer, 2015.

[61] Hoo Chang Shin, Holger R. Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M. Summers. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE Transactions on Medical Imaging*, 35(5):1285–1298, 2016.

[62] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[63] Deepak Soekhoe, Peter van der Putten, and Aske Plaat. On the Impact of Data Set Size in Transfer Learning Using Deep Neural Networks. *Advances in Intelligent Data Analysis XV: 15th International Symposium, IDA 2016, Stockholm, Sweden, October 13-15, 2016, Proceedings*, pages 50–60, 2016.

[64] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(4):427–437, 2009.

[65] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[66] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR*, 2014.

[67] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[68] Nima Tajbakhsh, Jae Y. Shin, Suryakanth R. Gurudu, R. Todd Hurst, Christopher B. Kendall, Michael B. Gotway, and Jianming Liang. Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? *IEEE Transactions on Medical Imaging*, 35(5):1299–1312, 2016.

[69] Lisa Torrey and Jude Shavlik. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 1:242, 2009.

[70] Bram van Ginneken, Arnaud A. A. Setio, Colin Jacobs, and Francesco Ciompi. Off-the-shelf convolutional neural network features for pulmonary nodule detection in computed tomography scans. In *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*, pages 286–289. IEEE, 2015.

[71] Juan Wang, Zhiyuan Fang, Ning Lang, Huishu Yuan, Min Ying Su, and Pierre Baldi. A multi-resolution approach for spinal metastasis detection using deep Siamese neural networks. *Computers in Biology and Medicine*, 84(March):137–146, 2017.

[72] Karl Weiss, Taghi M. Khoshgoftaar, and Ding Ding Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, 2016.

[73] Jelmer M Wolterink, Tim Leiner, Max A Viergever, and Ivana Išgum. Generative adversarial networks for noise reduction in low-dose ct. *IEEE transactions on medical imaging*, 36(12):2536–2545, 2017.

[74] Lina Xu, Giles Tetteh, Mona Mustafa, Jana Lipkova, Yu Zhao, Marie Bieth, Patrick Christ, Marie Piraud, Bjoern Menze, and Kuangyu Shi. W-net for whole-body bone lesion detection on ˆ{68} ga-pentixafor pet/ct imaging of multiple myeloma patients. In *Molecular Imaging, Reconstruction and Analysis of Moving Body Organs, and Stroke Imaging and Treatment*, pages 23–30. Springer, 2017.

[75] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[76] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.