FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Evaluation of Machine Learning Frameworks on Tuberculosis Classification of Chest Radiographs

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Katharina Unger
Matrikelnummer 01325652

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.tech. Eduard Gröller
Mitwirkung: MSc. David Major
Dipl.-Math. Dr. Katja Bühler

Wien, 28. November 2017

Katharina Unger	Eduard Gröller

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Evaluation of Machine Learning Frameworks on Tuberculosis Classification of Chest Radiographs

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Katharina Unger
Registration Number 01325652

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.tech. Eduard Gröller
Assistance: MSc. David Major
Dipl.-Math. Dr. Katja Bühler

Vienna, 28th November, 2017

Katharina Unger

Eduard Gröller

Erklärung zur Verfassung der Arbeit

Katharina Unger
Anton Hermann-Straße 8a, 2514 Traiskirchen

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, 28. November 2017

Katharina Unger

Danksagung

Ich möchte mich bei meinem Betreuer Msc. David Major für seine ausgezeichnete Beratung und Unterstützung bei dieser Arbeit und für die Bereitstellung nützlicher Materialien bedanken.
Außerdem möchte ich mich bei Dipl.-Math. Dr. Katja Bühler bedanken, für die Möglichkeit, meine Arbeit am VRVis Zentrum für Virtual Reality und Visualisierung Forschungs-GmbH zu schreiben und einen Einblick in die praktische Anwendung von Forschung in der Arbeitswelt erhalten zu haben.

Acknowledgements

I would like to thank my supervisor MSc. David Major for his great guidance and support for this thesis and for providing useful materials.
Furthermore, I want to thank Dipl.-Math. Dr. Katja Bühler for the possibility to write my thesis at the VRVis Zentrum für Virtual Reality und Visualisierung Forschungs-GmbH and provide me an insight into the practical application of scientific research.

This work was funded by the Competence Centre VRVis. VRVis is funded by BMVIT, BMWFW, Styria, SFG and Vienna Business Agency in the scope of COMET - Competence Centers for Excellent Technologies (854174) which is managed by FFG.

Kurzfassung

In dieser Arbeit wurden verschiedene state-of-the-art Machine Learning Frameworks zur Klassifikation von Lungenröntgen implementiert und evaluiert. Die Röntgenbilder sollen in Bilder klassifiziert werden, die Tuberkulose zeigen oder gesunde Röntgenbilder. Sowohl traditionelle Merkmalsextraktion als auch verschiedene Deep Learning Frameworks wurden ausgeführt. Für die Deep Learning Experimente wurden verschiedene, öffentlich verfügbare, Architekturen, in zwei verschiedenen Versuchen, verglichen. Der erste Versuch mit Deep Learning umfasste, Convolutional Neural Networks, welche zuvor an einem anderen Datensatz trainiert wurden, zu verwenden, um Merkmale aus den Lungenröntgen zu extrahieren. Diese Merkmale wurden dann seperat klassifiziert. Beim zweiten Versuch wurden wieder, zuvor trainierte, Convolutional Neural Networks verwendet. Diese wurden vorsichtig erneut mit dem neuen Datensatz der Lungenröntgen trainiert. Die Ergebnisse der verschiedenen Frameworks wurden zusammengefasst und in Tabellen präsentiert, sowie evaluiert.

Abstract

In this thesis different state-of-the-art machine learning frameworks were implemented and evaluated on chest radiographs to classify them into tuberculotic or healthy radiographs. Traditional explicit feature engineering was performed, as well as different deep learning approaches were applied. For the deep learning experiments different publicly available architectures were compared in two different tasks. The first task with deep learning was to use a Convolutional Neural Network, already trained on a different task, to extract features of the chest radiographs. These features were then classified separately. The second experiment was to use a Convolutional Neural Network, again pretrained on a different task, and train this network carefully again on the chest radiographs. The results of the different frameworks were summarized, evaluated and presented in tables.

Contents

CHAPTER 1

Introduction

1.1 Motivation

According to estimates of the annual Global Tuberculosis Report from the WHO [WHO16] from 2016, there were 1.4 million deaths by the tuberculosis disease in 2015. In 2015 tuberculosis was one of the top 10 causes of death [WHO16]. Another 0.4 million tuberculosis deaths occured among HIV-positive people. 10.4 millionen new tuberculosis incidents were estimated for 2015. Of these, 56% were men, 34% were women and 10% were children. Studies conducted before medication was available showed that up to 70% of tuberculosis victims died within the next 10 years.
Thanks to fast diagnosis and proper treatment, most tuberculosis cases can be cured. Success rates of at least 85% were reported in the cases of which tuberculosis is not drug resistant.
Since tuberculosis is so widespread and well treatable, it became one of the Sustainable Development Goals (SDGs) [Uni16] of 2030, issued by the United Nations to end the epidemic of tuberculosis. To fulfill this goal the WHO developed the End TB Strategy which is documented in their Global Tuberculosis Report [WHO16] from 2016. They aim to reduce deaths caused by tuberculosis by 90% from 2015 to 2030.
One of the main parts in this mission is the early diagnosis of tuberculosis of many people. Therefore, it is helpful to develop a method that can perform a rapid classification of pulmonary radiographs in healthy and tuberculotic lungs. This program can then be used in mobile X-ray screenings in areas heavily affected by tuberculosis. This enables patients to receive treatment at an earlier stage of the disease, which increases the chance of curing tuberculosis.
In this thesis we present different methods on the task of pulmonary tuberculosis classification of chest X-rays. The goal is to find out which of these methods provides the best results in terms of tuberculosis classification. Our focus is on pulmonary Tuberculosis, but different forms exist, which affect other organs than the lungs. These forms are

summarized under the term extrapulmonary tuberculosis [WHO16].

1.2 Problem Statement

The aim of this thesis is to evaluate how well the classification into healthy and tuberculotic lungs works, based on chest radiographs only. The framework should be able to classify different forms of pulmonary Tuberculosis depending on the available data for training. In order to achieve the best possible results, different machine learning frameworks were applied and evaluated. The classification of radiographs was chosen because it is the most commonly used type of medical imaging [Leu99] and it is the first step in Tuberculosis detection due to its quick availability. To give an example [Dia17], 54% of all diagnostic images made from march 2016 to march 2017 in England were radiographs. Therefore a lot of data exists that could be used to train and test the resulting frameworks. This framework should support doctors in classification and to make their workflow faster. It could be used in third world countries and with trained staff, to screen many people and deliver fast, reliable diagnosis to start the appropriate treatment.

1.3 Pulmonary Tuberculosis (TBC)

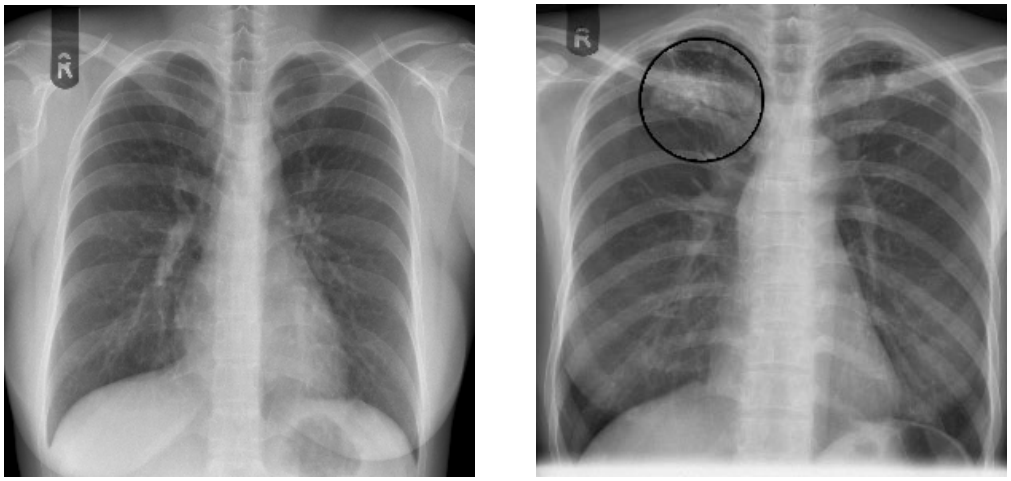

Figure 1.1: Chest radiographs showing a patient without TBC (left) and a patient with TBC (right). In the right radiograph manifestations of TBC are visible in the upper region of the right lung as nodular opacities inside the circle.

Tuberculosis is a disease with an infectious agent [Leu99] caused by the tubercle bacillus also called Mycobacterium tuberculosis. The transfer of the bacillus takes place by droplet infection through the air.
The epidemic of TBC is more prevalent in developing countries caused by the poor sanitary conditions compared to first world countries [Leu99]. As described by the WHO [WHO16] in 2015 60% of the new TBC incidents appeared in only six countries (India,

Indonesia, China, Nigeria, Pakistan, South Africa).
According to estimates of the WHO in their Global Tuberculosis Report [WHO16] from 2016, there were 1.4 million deaths caused by the tuberculosis disease in 2015. Another 0.4 million tuberculosis deaths were recorded among HIV-positive persons. 10.4 millionen new tuberculosis incidents were estimated for 2015. 56% of the patients were men, 34% were women and 10% were children.
To diagnose TBC bacteriologically the following tests exist [WHO16]:

- Sputum Smear Microscopy
  Sputum samples are being tested for bacteria under a microscope. This test only works for pulmonary TBC.
- Rapid Molecular Test
  The rapid molecular test gives better results than the sputum smear microscopy. It is recommended by the WHO for children and adults to diagnose pulmonary TBC and some types of extrapulmonary TBC.
- Culture Methods
  The WHO defines this test as the current reference standard. The disadvantage, however, lies in the duration of the evaluation, which takes up to 12 weeks.

There are also other tests for drug-resistant TBC.
Leung [Leu99] describes that chest radiographs make it easier to diagnose active or prior TBC. But the difference of these forms can only be diagnosed on the change of the radiographs over time. If there are no changes over 4-6 months in the images, the TBC infection can be considered as inactive. Chest radiographs have the advantage that they are quickly available compared to the reference standard defined by the WHO, which takes up to 12 weeks for analysis. Manifestations of TBC on radiographs can get detected in time and treatment can be started.
In his state-of-the-art report about pulmonary TBC Leung [Leu99] describes the differences of four forms of pulmonary TBC in chest radiographs depending on age, immune status and prior active TBC.

Primary TBC

The main characteristic of primary disease are enlarged lymph nodes (Lymphadenopathy) which occur in 83-96% of cases in children. The percentage decreases for older patients and can reach as low as 10-43%. Lymphadenopathy can be visible everywhere in the lung, but most frequently manifests at the right paratracheal and hilar stations.

Postprimary TBC

Most frequently and most characteristic for postprimary disease are parenchymal opacities at the upper lung lobes in 83-85% and at the top of the lower lobes in 11-14%.

Tuberculomas are round, sharply separated lesions with a size of 0,5-4,0 cm which appear in 3-6% of the parenchymal of postprimary TBC patients. Cavitations are visible in 40-45%. Calcified lymph nodes and fibrotic manifestations with an occurence rate of 20-40% indicate the primary presence of TBC.

Miliary TBC

In radiographs this form of TBC manifests itself in many small, 1-3mm, nodules distributed in the whole lung, which are present in about 30% of cases.

TBC in acquired immunodeficiency syndrome

The progress of the immunosuppression of HIV patients influences the manifestations of TBC. If the immune function is still nearly normal, it looks similar to the TBC radiographs of patients without HIV. In lungs of patients with HIV more Lymphadenopathy and less cavitation manifests.

To prevent TBC there is a vaccination [WHO16] that protects children from the infection of some forms of TBC, the Bacille-Calmette-Guérin vaccine, however it has no effect for adults.
The treatment of an existing active TBC is only possible with drugs. The current treatment recommended by WHO [WHO16] for TBC that is susceptible to the standard medication is fixed to 6 months. Other forms of TBC which are resistant to one ore more active substances need more expensive treatment, which takes about 9-12 months.

1.4 Methodological Approach

The methodological approach of this thesis is to evaluate different feature extraction methods to obtain the best result in terms of tuberculosis classification in chest radiographs. We want to compare traditional approaches with explicit feature engineering and classification to recent deep learning attempts on their performance in classification results and timing.
Firstly explicit feature engineering of different features is performed and then these features are classified with a Support Vector Machine (see section 4.2.1).
Secondly we use pretrained Convolutional Neural Networks as fixed feature Extractor (see section 4.2.2) and again classify the extracted features with a Support Vector Machine.
The last approach is to fine-tune a complete Convolutional Neural Network on the TBC classification task.

1.5 Structure of the Work

Chapter 2 covers the current state of the art in feature engineering and convolutional neural network approaches. Most of the discussed publications have the same goal

as this thesis, namely classification of pulmonary tuberculosis in chest radiographs. Additionally, to get a better understanding of the results, the human performance on reading radiographs is described. In Chapter 3, a short introduction to the important datasets relevant to this thesis, can be found. The available TBC dataset for this thesis to perform classification is described. Furthermore we describe the big labeled ImageNet dataset which made it possible to use pretrained deep networks with small datasets. In Chapter 4 the concepts needed for the three implemented approaches of this thesis are briefly described. These are feature engineering, support vector machines and convolutional neural networks. In addition, different methods on the application of convolutional neural networks and the most popular and successful architectures are mentioned. The technology consisting of hardware and software used in this thesis is described in Chapter 5. Besides that, the implementation of the three approaches is explained. Each approach is structured into the steps of preprocessing, feature extraction and classification. The results of the performed experiments are documented in Chapter 6. For each approach, the results are presented and a comparison of the best results of all experiments is provided. The last chapter covers the critical reflection of the implemented approaches. It draws a comparison with related work and the open issues are discussed.

CHAPTER 2

State of the art

This chapter provides a brief overview of the current methodology relevant in this thesis to identify tuberculosis in chest radiographs. Since different methods have been evaluated, the chapter is structured accordingly to the methods used. Firstly feature engineering is introduced, which aims to provide an insight into explicit feature extraction, like edge detection and shape detection. In another step, different deep learning approaches will be examined. The last part of this chapter will give an overview of human performance. This information enables us to make a direct comparison of the performance between experienced humans and the different presented methods.

2.1 Feature Engineering

Most of the available publications concentrate on individual areas of chest radiographs, for example [AGM98], [vGSL06], [DPU+09]. Apart from these works, there are also some publications which are particularly concerned with the classification of TBC.

The method of Jaeger et al. [JKC+14] starts with the segmentation of the lung, by using Graph cut and a predefined lung model. The resulting lung fields are then used for feature extraction in terms of intensity, gradients, shape, edges and other features. Afterwards, classification of the features is done by using a Support Vector Machine (see section 4.2.1). For classification the accuracy is between 78-82,5%, depending on the used dataset.
Melendez et al. [MvGM+16] introduced their approach on TBC classification in chest radiographs using a combination of Multiple Instance Learning and Active Learning. Their approach starts with lung segmentation and feature extraction of texture features based on the intensity distribution. After the initial classification of the training set, the most useful images in positive bags are separated into image regions, which are then classified by an expert. With their approach they reach an Area Under Curve (AUC) of

up to 88%.
The publication of Hogeweg et al. [HSM+15] start with segmentation of the lungs like Jaeger et al. [JKC+14] and Melendez et al. [MvGM+16]. In their approach they place the focus on the irregularity of the manifestations of TBC. Different subsystems are used, which classifiy shape and texture features to get different subscores. The combination of these subsystems to form a total score, is an attempt to reach good generalization. The best AUC reached by this combination is 86%.

2.2 Convolutional Neural Networks (CNN)

The breakthrough of the CNNs in visual computing was in 2012 with the so-called AlexNet from Krizhevsky et al. [KSG12]. It won the ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC). Since then, researchers attempted to develop different methods to make use of CNNs. Besides the opportunity to build your own network it is possible to make use of existing architectures (see section 4.2.2 for more details).
This section covers approaches on untrained networks as well as networks trained on the ImageNet dataset (see section 3.2) which were fine-tuned or used as fixed feature extractor. More details on these methods are available in Chapter 4.

Shin et al. [SRG+16] compared a few different networks, mainly AlexNet [KSG12] and GoogLeNet [SLJ+15] with each other on the task of interstitial lung disease detection on computed tomography. Tests on the untrained networks and fine-tuning of the pretrained networks were performed. The training data consists of 905 slices from 120 patients. They achieved an accuracy of 74% on AlexNet and 75% on GoogLeNet by training the models from scratch with this dataset. They stated that with such a small dataset it is hard to train big networks. Their results achieved on fine-tuning the pretrained networks were for AlexNet 86,7% accuracy and for GoogLeNet 90,2%. Additionally, fine-tuning was performed on OverFeat from Sermanet et al. [SEZ+13] with an accuracy of 87,7% and on VGG16 from Simonyan et al. [SZ15] with an accuracy of 90%.
The approach of Hwang et al. [HKJK16] is also based on AlexNet [KSG12] for the goal of TBC classification in chest radiographs. They also compared the training of the untrained network with fine-tuning of the pretrained network. Their dataset consists of 10848 images from which 7020 are normal and 3828 have manifestations of TBC. The data was augmented using random cropping from 520x520 images to 500x500 resolution and mirroring. AlexNet is trained on input data of size 227x227, so Hwang et al. added an extra convolutional layer for feature extraction, to make use of their input size of 500x500. By training the AlexNet architecture from scratch with random initialized weights, they could reach an AUC of 82,8% and an accuracy of 78,8%. With fine-tuning, they reached an AUC of 96,7% and an accuracy of 90,5%.
The paper of Lakhani et al. [LS17] is again making a comparison of AlexNet [KSG12] and GoogLeNet [SLJ+15], with untrained weights and pretrained weights, for the clas-

sification of TBC in chest radiographs. They had 1007 images, from which 150 were used for testing (75 TBC positive, 75 healthy), 685 for training and 172 for validation. Data augmentation was done by resizing to 256x256, converting to Portable Network Graphics format, random cropping to size 227x227, mean subtraction and mirroring. With additional augmentation of rotations of $90°, 180°, 270°$ and contrast limited adaptive histogram equalization they were able to further improve their untrained models. Their best results for untrained networks were an AUC of 90% with AlexNet and an AUC of 88% with GoogLeNet. An ensemble on the best performing fine-tuned GoogLeNet and AlexNet reached an AUC of 99%.

Another method of making use of Convolutional Neural Networks is to apply already trained networks as fixed feature extractors.

Shin et al. [SRG+16] also based an approach on this method in their paper. The only network, this approach was applied to, was AlexNet [KSG12]. The extracted features could be classified with an accuracy of 76%.
Razavian et al. [RASC14] used the OverFeat network [SEZ+13] pretrained on ImageNet [JWS+09] for their experiments. Their dataset was the Pascal VOC 2007 dataset [EEZ+06], consisting of more than 10000 images and 20 classes, which were augmented using cropping and rotation. The output of the first layer was used as feature vector with dimension of 4046 for classification. The classification of the feature vector was accomplished using a Support Vector Machine (see section 4.2.1). Their mean average precision for all classes was 73,9% without data augmentation and 77,2% with data augmentation.
The publication of Ginneken et al. [vGSJC15] uses the extracted features of the OverFeat network for pulmonary nodule detection in CT scans. Their dataset consists of 865 CT scans with 1147 pulmonary nodules. At first nodule candidates were extracted using a state-of-the-art nodule detection system. For every candidate and for every x,y,z axis 2D images were extracted from the scans and rescaled to 221x221 resolution. This patches were used as input for the OverFeat network and the resulting 4096 dimensional feature vector was then classified with a Support Vector Machine (see section 4.2.1). The state-of-the-art nodule detection system reached a sensitivity of 68% and could be improved with the additional usage of OverFeat as feature extractor to 71%.

2.3 Human Performance

To gain an understanding about the quality of the performance of different machine learning approaches, we present results for the performance of humans who are trained in reading radiographs in this section.
Jaeger et al. [JKC+14] could get two radiologists for a second and third reading of the radiographs. Both radiologists were aware of the task of TBC detection in radiographs and read the images independently. The ground truth of the images was based on clinical and patient data, to which the radiologists had no access. After the first reading of

the images, the radiologists agreed in 84,8% of the cases. For the cases, in which they disagreed, they decided about the status of the patient together. This collaboration resulted in detection of all positive TBC cases, therefore a sensitivity of 100% and a specificity of 68,8%.
Maduskar et al. [MMA+13] had 4 clinical officers, who read the chest radiographs. These clinical officers had a 3-year diploma in medicine and are trained on the task of interpreting chest radiographs. No clinical information of the patients was provided. The images were classified with a score between 0–100, where the score expresses the confidence of the clinical officers in active TBC. A score bigger than 50 was interpreted as an abnormal appearance. The AUC for the clinical officers were 89%, 90%, 91% and 92%, the sensitivity 86%, 85%, 83% and 96% and the specificity 88%, 76%, 85% and 46%.

CHAPTER 3

Data

This chapter gives a general description of the sample data used in this thesis. First the TBC dataset is described. Additionally the ImageNet dataset [JWS+09] is introduced, because it is an important dataset in terms of Convolutional Neural Networks. Thanks to the big size of the dataset, it is possible to train deep Convolutional Neural Networks properly. These networks can then be reused on other tasks, with less available data, as described in section 4.2.2.

3.1 TBC Dataset

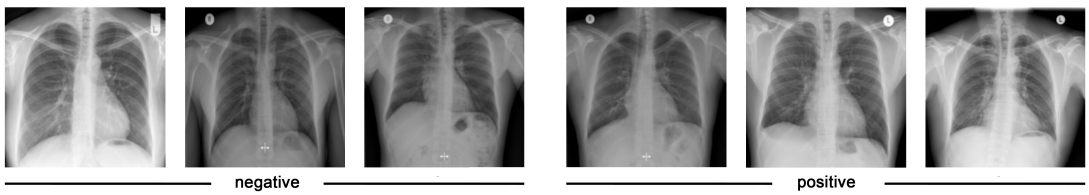

Figure 3.1: Example images from the TBC dataset. The three images on the left are healthy (negative) and the three images on the right are labeled as tuberculotic (positive).

The dataset used in this work has been provided by the industrial partner of this project. The data has been anonymized, so no clinical, gender or age information can be read from it. It consists of 1544 digital radiographs of the chest, provided in the DICOM standard [MEM02]. The size of the images ranges in width from 1832 to 2992 pixels and in height from 1787 to 2931 pixels. 877 images are labeled as healthy and the other 667 as tuberculotic. The radiographs were labeled, based on the results of Sputum Smear Microscopy. The tuberculotic images depict different forms of TBC and are not restricted to one type of manifestation. We have a heterogeneous dataset containing progressive

stages of primary TBC as well as early stages and postprimary TBC. Example images of this dataset with healthy and tuberculotic lungs are provided in Figure 3.1.
85% of these images are used for training, the other 15% for testing, which results in 1312 images for training and 232 images for testing. The test set is identical for all implemented methods. For the methods including the training of Convolutional Neural Networks, 20% of the training set are used for validation of the trained model. This results in 1049 training images and 263 validation images and, like before, 231 test images.

3.2 ImageNet

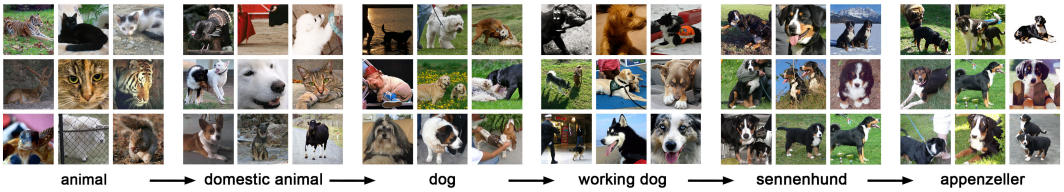

Figure 3.2: Visualizations of a branch form the root to the leaf of ImageNet. For each category, 9 randomly sampled images are presented.

The ImageNet [JWS+09] dataset is important in terms of Convolutional Neural Networks. It consists of many natural images, which are used for the training of big Convolutional Neural Networks.
The structure of the dataset is based on the WordNet structure [Fel98]. In the publication from 2009 [JWS+09] it is stated that the dataset consists of 3.2 million images in 12 subtrees with 5247 different categories. Each category holds 600 images on average. Current numbers from the ImageNet Homepage [JWS+09] state that the dataset holds 14,197,122 images in 21841 categories.
To obtain better diversity, the labeled objects in the images can be occluded and the images can include background clutter and show different appearances, positions, view points, and poses.
The classification of the images was fulfilled by humans with the use of the service of Amazon Mechanical Turk (AMT). This service is an online platform, which pays users for the completion of provided tasks. The average precision on the labeling of the images is 99,7%.
With the creation of this dataset the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was introduced in 2010 and has been held annually since then. Its aim is to judge different algorithms for object localization and object detection.
In this thesis we did not use the image data itself, but the weights of Convolutional Neural Networks trained on this dataset.
In figure 3.2 an example branch is visualized with randomly selected images.

CHAPTER 4

Methodology

This chapter describes the concepts which are relevant for this thesis, starting with the description of the method of Feature Engineering and the extracted features in this work. Afterwards Support Vector Machines as a classification technique are presented. Then Convolutional Neural Networks are explained, as well as different methods on how to apply such networks. Additionally different well-known architectures are introduced. These architectures are AlexNet from Krizhevsky et al. [KSG12], GoogLeNet from Szegedy et al. [SLJ+15], VGGNet from Simonyan et al. [SZ15] and ResNet from He et al. [HZRS16].

4.1 Feature Engineering

Feature Engineering is about explicitly extracting features from an image. The approach implemented and evaluated in this thesis for feature engineering is oriented along the implementation of feature set A from Jaeger et al. [JKC+14].
Before feature extraction the input data needs to be preprocessed. Common preprocessing steps are resizing every sample to the same size and perform standardization of the intensity for each sample. We resized the images using spline-interpolation to a resolution of 256x256 pixels and then rescaled the intensity to a range of $[0, 1]$.
Additionally segmentation of the important region of the image can be performed. First a model of the average lung is computed. The lung is then segmented by applying a graph cut approach by minimizing the objective function:

$$E(f) = E_d(f) + E_s(f) + E_m(f)$$

where E_d stands for the region properties, E_s the boundary properties and E_m the lung model properties, as described by Jaeger et al. [JKC+14].
In the next step, the desired features can be computed. Useful features are edges, shape, intensities, and other texture features.

The concepts which were used in this thesis for feature extraction will be briefly described in the next paragraphs. For each feature descriptor we calculate a histogram. Each bin of the histogram represents one measurement of the feature. Combining all features from every descriptor results in our feature vector. This vector will then be classified with a Support Vector Machine. We use 32 bins per histogram, because Jaeger et al. [JKC+14] state that empirical experiments showed that this number of bins shows good results.

The **Gradient Magnitude** of an image is calculated by filtering the image with Gaussian derivatives. The gradients show the change of intensity in an image. A gradient consists of two components, the horizontal gradient and the vertical gradient:

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$

The gradient magnitude is the combination of the horizontal and vertical component:

$$g = \sqrt{g_x^2 + g_y^2}$$

This characteristic is useful for edge detection. As described before, a histogram of the filtered image is computed afterwards.
For texture classification the **Local Binary Pattern** descriptor, first described by Ojala et al. [OPH94], is used. The intensity of a pixel is compared to the intensity of its neighbours. If the intensity of a neighbouring pixel is greater than the intensity of the center pixel, '0' is written as the intensity of the neighbour, else '1'. After binary labeling, a histogram for classification is computed, as part of the Local Binary Pattern algorithm.
Histogram of Oriented Gradients (HOG), first described by McConnell [McC86], is a descriptor for detection of objects in images. The image is divided into cells. A histogram of the gradient directions is then computed for each cell. The features for classification result from the combination of these histograms.
Shape and curvature features were extracted with the Frangi filter based on the Hessian matrix [FNVV98]. The eigenvalues λ_1 and λ_2 of the Hessian are extracted to calculate the shape as follows:

$$s = tan^{-1}\left(\frac{\lambda_1}{\lambda_2}\right)$$

and the curvature:

$$c = tan^{-1}\left(\frac{\sqrt{\lambda_1^2 + \lambda_2^2}}{1 + I(x,y)}\right)$$

where $I(x,y)$ is the intensity of the pixel (x,y).
An **Intensity histogram** of the pixel intensities is computed.

With 6 types of histograms of features - Gradient Magnitude, Local Binary Pattern, HOG, Shape, Curvature and Intensity - and 32 bins per histogram, we get a total of 6*32 = 192 features for classification.

4.2 Classification

In this section two approaches to solve the classification problem are presented.
The classification problem is the task of finding the correct label from a predefined set of categories to an input image. We want to find a function $f : X \mapsto Y$ such that the input images $x_i \in X, i = 1, ..., N$ are mapped to the corresponding labels $y_i \in 1, ..., M$.

In the next paragraphs the Support Vector Machine and Convolutional Neural Networks are presented. Both are supervised approaches in solving the classification problem.

4.2.1 Support Vector Machine (SVM)

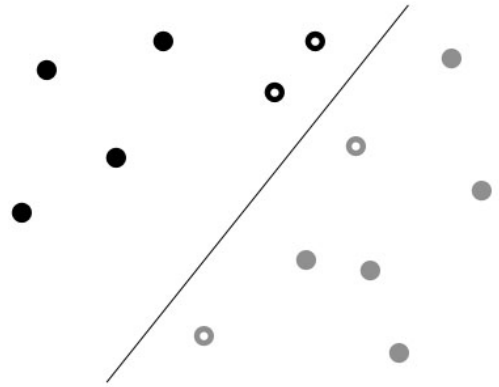

Figure 4.1: Linear classified 2D dataset. The support vectors are visualized with a white dot in the middle.

The Support Vector Machine [Bur98] is a supervised learning algorithm for classification of features.
The linear SVM solves the classification problem as a linear function

$$f(x) = \boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b} \tag{4.1}$$

where $\boldsymbol{W}$ is a matrix, referred to as weights, and $\boldsymbol{b}$ is a bias vector. For a binary classification problem, as in this thesis, applying the signum function to the linear function (4.1)

$$y = \hat{f}(x) = sgn(\boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b})$$

leads to a mapping of the output to 1 or -1. This output can then be interpreted as class labels.
SVMs are looking for a separation of the data in such a way that the distance between the datasamples of the different classes is as large as possible. For a linear SVM this means finding a separating hyperplane, for a non-linear SVM the algorithm is looking for a decision surface.

The name of the SVM results from the so-called support vectors, which are datapoints that directly influence the decision boundary. If one support vector is removed, the resulting decision boundary will be different. If all datapoints except the support vectors are removed, the boundary stays the same. The support vectors additionally have the characteristic that they are the closest datapoints to the separating hyperplane or surface. To obtain non-linear decision surfaces a kernel function was introduced [GBC16], which enabled more efficient calculation. This is done by rewriting the linear function as:

$$\boldsymbol{W} \cdot \boldsymbol{x} + b = b + \sum_{i=1}^{m} \alpha_i \boldsymbol{x}^\intercal \boldsymbol{x}^{(i)}$$

where α is a vector of coefficients with mostly zeros. It results from learning which training samples contribute to the decision surface. $\boldsymbol{x}$ is then replaced with a feature function $\phi(\boldsymbol{x})$. The kernel function then replaces the dot product

$$k(\boldsymbol{x}, \boldsymbol{x}^{(i)}) = \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{x}^{(i)})$$

which results in the function

$$f(\boldsymbol{x}) = b + \sum_{i} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}^{(i)}).$$

In practice the parameters for SVMs need to be predefined. To find the best parameters Grid Search can be processed. For a given set of parameters the Grid Search algorithm tests all possible combinations of parameters on the dataset. These combinations are then evaluated and the best parameter set is selected for classification.

4.2.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks [LKJ] are a special type of Neural Networks. Neural Networks consist of stacked layers, beginning with an input layer, several hidden layers and an output layer. Figure 4.2 shows a visualization of this structure. The input is a single vector. A hidden layer is made of multiple neurons where each neuron of one layer is connected to each neuron of the previous layer, but there are no connections between neurons inside one layer. A neuron has weights and biases, as seen in function (4.1), which can be trained. The output layer is a fully connected layer and outputs the class probabilities.

CNNs usually take RGB images as input. However, the input must not necessarily be an image, it just needs to be a matrix. To process this input, the mathematical operation of convolution is used. The fully-connected structure of Neural Networks is designed for processing a single vector [GBC16] and would be too complex for an approach on images. Therefore CNNs have sparse interactions between the neurons, by employing kernels which are smaller than the input.
Just like regular Neural Networks, CNNs consist of several stacked layers. In figure 4.3 the

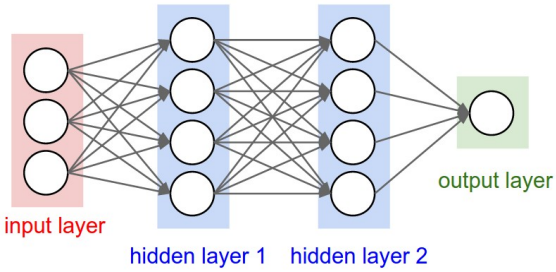

Figure 4.2: Visualizations of the structure of a Neural Network by Li et al. [LKJ].

architecture of the well-known LeNet CNN by LeCun et al. [LBBH98] can be seen. The most important type of layer is the convolutional layer. In this layer the convolution of the input image with different filters is computed. A CNN contains multiple convolutional layers, where each layer has filters, whose complexity depends on the location of the layer inside the network. Filters of early layers detect low level features like edges and shapes, whereas later filters are able to detect high level features corresponding to the given input data. The results of the convolutions of these filters with the input image are then combined to receive the output of the network. The output is a vector with class probabilities, where the size of the vector depends on the number of classes.

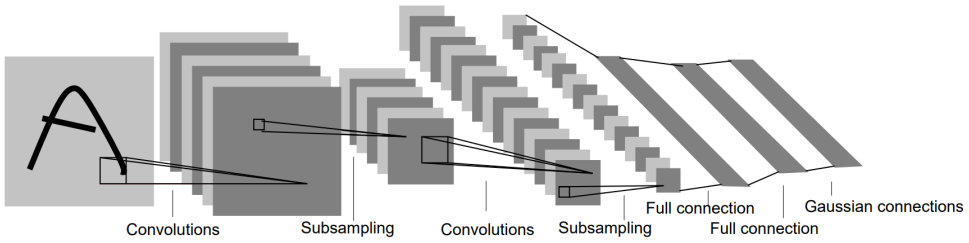

Figure 4.3: Visualization of the LeNet architecture by LeCun et al. [LBBH98].

Besides the convolutional layer there exist several types of layers with different purposes. In the tutorial of the Stanford University by Li et al. [LKJ] five types of layers are presented:

- Input Layer
  The Input Layer of CNNs is the input with matrix structure. In most applications this are images with RGB color channels.

- Convolutional Layer
  In this layer the convolution of the input volume with the weights of the neurons is carried out. The weights are parameters which are learned during the training phase of the network.

- Rectified Linear Unit (ReLU) Layer
  Per pixel an activation function is applied. For example $max(0, x)$ would set negative entries to zero. As this is a fixed function, ReLu Layer do not provide trainable parameters.

- Pooling / Subsampling Layer
  The Pooling Layer is responsible for downsampling the input in width and height, but not in depth. This step is useful to smoothen the image and therefore reduce the influence of noise. Just like the ReLU layer, this layer has no trainable parameters.

- Fully-Connected Layer
  The last layer of CNNs, is the Fully-Connected layer. As mentioned before, in case of CNNs it outputs the class scores and every neuron in this layer is connected with each neuron of the previous layer. This layer implements computation using trainable parameters.

Li et al. [LKJ] additionally describe the typical CNN architecture as repeatedly stacked Convolutional(Conv)-ReLU layers. They can optionally be followed by Pooling layers (Pool). The last layer always is a Fully-Connected layer. The following line shows a typical pattern presented by Li et al.:

Input → [[Conv → ReLU]*N → Pool?]*M → [FC → ReLU]*K → FC

The ? means to add a layer optionally and the $*$ means to repeat the term as often as specified. N, M and K can be chosen individually, but they need to be natural numbers including 0.

Training of CNNs

To train a CNN several building blocks need to be considered. These building blocks and the training process itself will be described in the next paragraphs.

The **score function** $f(x)$ (4.1) gets raw data as input and outputs the class probabilities. The function has a set of parameter, which can be controlled, or in terms of CNNs, they can be trained. The goal is to adapt the weights such that the class probabilities match the ground truth labels as closely as possible.
To obtain this, the **loss function** is needed. Its task is to determine how good the prediction matches the ground truth. High loss indicates poor classification, while low loss shows good classification.
The most popular loss function used with CNNs is the **cross-entropy loss** with the form:

$$L = -\sum_{i=1}^{N}(y_i \; log \; q_i)$$

where q is estimated using the **Softmax function**. The Softmax function outputs the class probabilities between 0 and 1, which sum up to 1.
The process, of finding the weights, which minimize the loss function is an optimization problem.
The gradient of the loss function indicates the best direction to change the weights. **Gradient Descent** is the process of regulary performing a parameter update by calculating and evaluating the gradient.
The issue with gradients is that we do not know how far we have to go into the direction specified by the gradient. To make any progress we have to set some step size, which we use to carefully follow the direction of the gradients. This step size is often referred to as **Learning Rate**. If we choose a step size which is too small, only small progress is made. If it is too big, we could step over the optimum.
For better performance, the gradient descent and the parameter update are not performed for every training sample. It is only performed for batches of training samples.
Backpropagation is the process of calculating the gradients through recursive use of the chain rule layer by layer of a CNN.

Before the training of a CNN, the training data needs to be preprocessed. The preprocessing of the input data consists of the standardization to get the same range for every data sample. Additionally, as CNNs are trained for an explicit input size, the input needs to be resized or cropped to that specified size.
Data Augmentation is as important as the preprocessing step. It helps to find a model, which provides good generalization for the input data. The augmentation is only applied on the training set. The test set remains unchanged except for the preprocessing. Common augmentations of the input are cropping, translations, rotations and flips.
If image cropping is performed, a window of the size which is taken as input of each network is cut out of a bigger image and used for training. Figure 4.4 shows an example of cropping 5 different images out of one bigger image.

The training process of a CNN follows a pipeline.
A batch of training samples is passed through the network to obtain class probabilities. With these probabilities and the labels of the images, the loss is calculated. Backpropagation is then used to calculate the gradient of the loss and afterwards the parameter update is performed. These steps are repeated until the desired results are obtained.

Different techniques exist to apply CNNs for a specific task. The next subsection describes the use of untrained networks, also referred to as training them from scratch. Afterwards different methods for Transfer Learning are presented. Additionally some of the most popular CNN architectures are introduced.

Training From Scratch

The parameters of a CNN are initialized using Gaussian distribution with a mean of 0 and a standard deviation of 0.01. Training an untrained network from scratch can

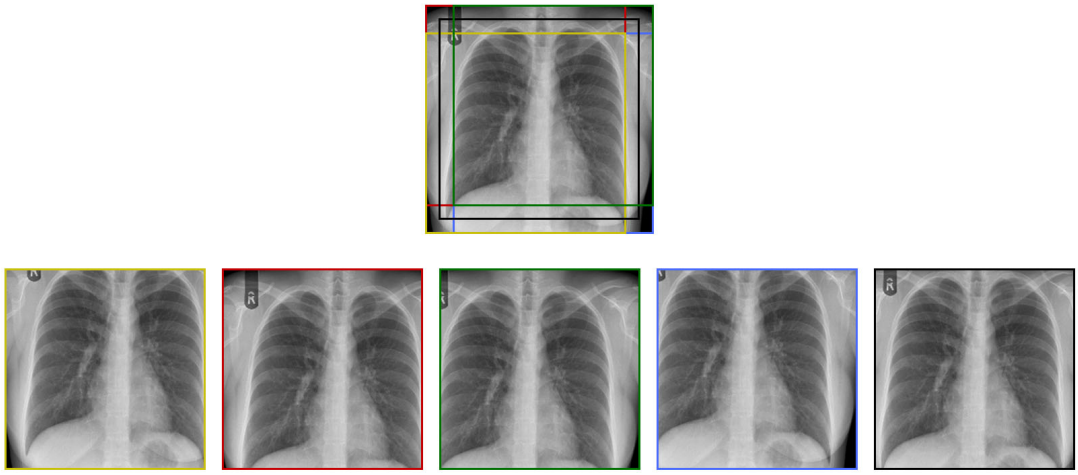

Figure 4.4: Image cropping: crop images form bottom left (yellow), top left (red), top right (green), bottom right (blue) and center (black).

be a hard task. For training deep networks like InceptionV3 (see section 4.2.2) a big dataset is required. One dataset which is big enough to train deep networks is ImageNet (see section 3.2). Training a deep CNN on ImageNet takes weeks on multiple GPUs [LKJ]. But not every classification task is similar to images of the ImageNet. If the new, different dataset is not big enough to train a network from scratch, it can be beneficial to use pretrained CNNs and transfer the learning from one datasets to a new, different dataset. This method is described in the next section.

Transfer Learning

Transfer Learning [YCBL14] is accomplished by using pretrained networks. Its aim is to use the trained parameters of one task for another, new, different task. For most well known architectures (see section 4.2.2) the weights of the CNNs, trained on different datasets, are publicly available. These networks are then referred to as pretrained networks. Most of the available weights have been calculated by training on ImageNet (see section 3.2). Pretrained CNNs can then be used as either a Fixed Feature Extractor or the weights can be fine-tuned. Transfer learning is useful due to the fact that CNNs learn more general, low-level features like edges and shapes in early layers and the more task specific features in later layers.

Fixed Feature Extraction

Using a CNN as a Fixed Feature Extractor is often also referred to as using off-the-shelf features. For this task a pretrained network is used. As input the data of a specific, different, dataset is used. To get class possibilities for the new classes, the network needs to be modified.
For classification one option is to exchange the last fully connected layer of the used

network with a layer corresponding to the number of classes, required for the new task. This new fully connected layer then needs to be freshly trained on the new dataset.
Another possibility is to completely remove the last fully connected layer, then extract the features for all data and in the end classify these feature vectors with a SVM (see section 4.2.1).

Fine-Tuning CNNs

Fine-Tuning networks means to train the weigths of a pretrained network again, but with a lower learning rate for the reason of classifying a different dataset. It has been shown that more general features, like edges and shapes are learned in every network in early layers [YCBL14].
To fine-tune the weigths of the more specific layer for the new dataset can therefore yield better results. All layers can be fine-tuned or only the later specific layer, this depends on the similarity of the new dataset to the initial training dataset.

CNN Architectures

A wide variety of different CNN architectures with different characteristics exist. This section provides a short description of the architectures used in this thesis in terms of classifying a dataset into healthy and TBC images.

The **AlexNet** architecture from Krizhevsky et al. [KSG12] is the winner of the ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC12) and is visualized in figure 4.5. This model brought the breakthrough of CNNs in visual computing. The network consists of five convolutional and three fully-connected layers with a total of 60 million parameters.

The **GoogLeNet** architecture form Szegedy et al. [SLJ+15] won the ILSVRC14. They developed a module called 'Inception' which represents a little network inside the network. The Inception module could reduce the number of paramaters to 12 times less than AlexNet. The architecture consists of several stacked Inception modules. GoogLeNet is 22 layers deep. Several improvements resulted in several versions following GoogLeNet. The improved version of GoogLeNet used in this thesis is InceptionV3 from Szegedy et al. [SVI+15]. The architecture from InceptionV3 can be seen in figure 4.6.

The **VGGNet** by Simonyan et al. [SZ15] is another approach, which attempts to increase the performance by increasing the depth of the network.
As mentioned before, low-level features can be found in earlier layers and specific features in later layers. This means that with increasing depth even more complex features can be learned.
The network took part in the ILSVRC14, just like GoogLeNet, which was able to outperform the VGGNet. The VGGNet resulted from tests in increasing depth between 11 to 19 layer. The best performing models, VGG16 with 16 layers and VGG19 with 19 layers were published. With the deeper models, the number of parameters increased to

138 million parameters in VGG16 and 144 million parameters in VGG19. In this thesis VGG16 is used, but VGG19 is visualized in figure 4.7.

The **ResNet** architecure by He et al. [HZRS16] won the first place at the ILSVRC15. As complexity increases with the depth of the network, He et al. were trying to improve learning by introducing skip connections and building deeper models. With a depth of up to 152 layers it is 8 times deeper than VGG19. They state that their network is less complex and that they can gain accuracy from the increased depth. In this thesis we use ResNet50 with a depth of 50 layers and in figure 4.8 ResNet34 with 34 layers is visualized.

Figure 4.5: Architecture of AlexNet visualized by Cohen [Coh].

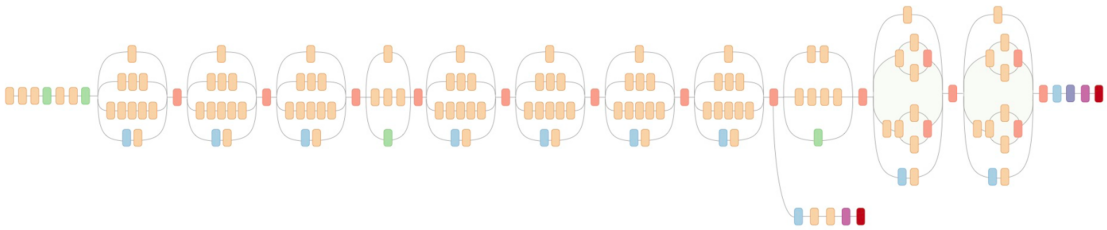

Figure 4.6: Architecture of InceptionV3 visualized by Shlens [Shl].

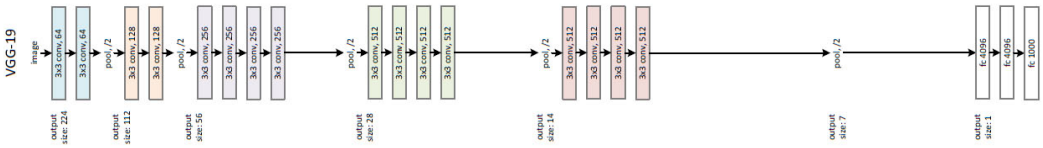

Figure 4.7: Architecture of VGG19 visualized by He et al. [HZRS16].

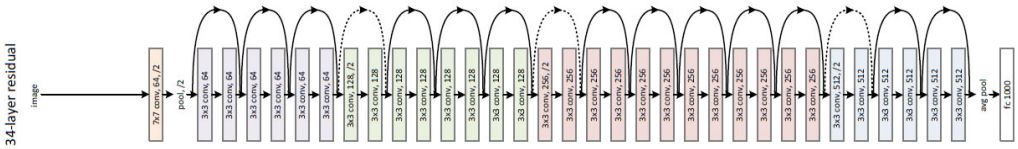

Figure 4.8: Architecture of ResNet34 visualized by He et al. [HZRS16].

CHAPTER 5

Implementation

In this chapter the implementation of the different approaches is explained. Starting with the technological setup, the used hardware is described. Additionally the programming language and valuable frameworks for deep learning and scientific computing are mentioned. Then, the implementation of the concepts presented in Section 4 and their interaction is described. This description is structured according to the different experiments performed, which are Feature Engineering, Fixed Feature Extraction, and Fine-Tuning.

5.1 Hardware

The Feature Engineering and Fixed Feature Extraction experiments were performed on a Intel Core i5 CPU with 32GB RAM and a NVIDIA GeForce GTX 1060 GPU of 3GB memory. The third experiment - fine-tuning - was executed on a Intel Xeon i5 and an NVIDIA GeForce GTX TITAN X with 12 GB.

5.2 Software

As programming language Python v3.5.2 was used. The different Convolutional Neural Network architectures were built and trained with Keras v1.2.0 [Cho15], a Deep Learning library written in Python. Theano v0.9.0 [The16] was used as backend engine for Keras. Scientific computation was obtained with the SciPy [Oli07] packages. The core package NumPy v1.11.1 [VCV11] was used for numerical computation and the SciPy library v0.18.1 [JOP$^+$] provided numerical algorithms. For reading and processing the radiographs, provided as DICOM standard, the python package Pydicom v0.9.9 [Mas11] was utilized. The Scikit-learn library v0.17.1 [PV11] provided modules for machine learning.

5.3 Feature Engineering

For explicit feature engineering the classical pipeline of preprocessing, feature extraction and classification will be executed. The next sections describe the implementation of these steps, according to the task of TBC classification in chest radiographs.

5.3.1 Preprocessing

As a first step the input images were resized using spline-interpolation to a size of 256x256 pixels. Next the intensities are rescaled to a range of $[0, 1]$.

5.3.2 Feature Extraction

The extraction of the features starts with the segmentation of the lung using an average lung model and a graph cut approach as described in Subsection 4.1. From the segmented lungs, the features were computed. These are histograms for intensities, gradient magnitude, local binary pattern, histogram of oriented edges and shape and curvature descriptors computed for each lung image.
To compute the Gaussian gradient magnitude, $\sigma = 1.0$ is used. Local binary pattern is implemented using radius $r = 1$ and the number of neighbour set points is 16. For the histogram of gradients a setting with 8 orientation bins, cell size (4,4) and (1,1) cells in each block is applied.
These features are then arranged in a matrix and saved for the classification. These steps follow the pipeline described in Section 4.1.

5.3.3 Classification

For classification SVMs (see section 4.2.1) are used. For a gain in performance, Grid Search is performed to obtain the best parameters according to the features. This can be very time consuming, if a lot of parameters are tested, since all combinations are tried. Therefore only 20% of the training set were used to perform Grid Search on. After the best parameters were found, these parameters were used to train the SVM from the beginning on the training data. The parameter used for Grid Search can be found in Table 5.1.

Parameter	**Values**
Kernel	linear, poly, rbf, sigmoid
Degree	$2, 3, 4, 5$
C	$2^{-7}, 2^{-5}, 2^{-3}, 2^{-1}, 2^{1}, 2^{3}, 2^{5}, 2^{7}, 2^{9}, 2^{11}, 2^{13}$
Gamma	$2^{-15}, 2^{-13}, 2^{-11}, 2^{-9}, 2^{-7}, 2^{-5}, 2^{-3}, 2^{-1}, 2^{1}$

Table 5.1: The parameters used to perform Grid Search.

The set of parameters which obtained the best classification is $[Kernel : linear, Degree : 2, C : 2^{-7}, Gamma : 2^{-15}]$.

5.4 Fixed Feature Extraction

The Fixed Feature Extraction is the first approach on CNNs. In this experiment the pipeline of preprocessing, feature extraction, and classification remains preserved.
We perform feature extraction on four different CNN Models, which are AlexNet, VGG16, InceptionV3 and ResNet50 (see Section 4.2.2). These four networks were chosen, because they set the state of the art in CNNs in the past years of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) and they are publicly available. All four models were pretrained on the ImageNet dataset.
Since the different models had different presettings for training the models, the steps, which are described in the following, can differ depending on the network.

5.4.1 Preprocessing

The first step was to resize the images. This was done according to the input size on which the networks were trained. AlexNet was trained on an input size of 227x227 pixels, VGG16 and ResNet50 with 224x224 pixels and InceptionV3 with 299x299 pixels. The intensity was rescaled to the range of $[0, 255]$ independent from the model. One important step in the preprocessing that needs to be considered is that the models were pretrained on the ImageNet dataset, which consists of RGB images. The TBC dataset contains only grayscale images with one channel. Therefore we tripled the single channel to imitate a RGB image. The influence of color normalization was tested. The values in Table 5.2 for color normalization are the values which were learned with the ImageNet dataset. These values were subtracted from the different color channels. The influence of color normalization on the results is documented in Chapter 6.

Channel	**Value**
R	$-123,68$
G	$-116,779$
B	-103.939

Table 5.2: Values that were subtracted from the images rescaled to an intensity of [0, 225], referred to as color normalization.

The test data is always directly resized to the according input size of the network and the intensity is rescaled as described. Additionally, if color normalization was performed on the training set, it was performed on the test set too. The steps described next, were only performed on the training data for generalization purposes and are referred to as data augmentation.

5.4.2 Feature Extraction

In CNNs the features are computed automatically. The only thing that needs to be done is to extract the computed features from the network. This step is the same, independent from the CNN architecture. As previously described, the last layer, a fully-connected

layer, of a network is responsible for classification. As we are not interested in classifying the chest radiographs in ImageNet classes, this layer is removed. The output of the layer before classifications are the features of an image. When the architecture is manipulated, such that features can be extracted, we enter our preprocessed and augmented training data and save the output of the network a matrix. For all different architectures only the last fully-connected layer was removed. Subsequently the features were extracted and classified as described in the next paragraph.

5.4.3 Classification

In this experiment the classification was performed with SVMs, just like in the task of feature engineering. The procedure stays the same. Grid Search is executed with the parameters written in Table 5.1 on 20% of the training set for each network. The TBC dataset is slightly unbalanced with 877 healthy and 667 tuberculotic images. Therefore as a last experiment the influence of setting class weights for the SVM according to that imbalance were performed.

5.5 Fine-Tuning

Fine-tuning a CNN was performed on the network, which yielded the best results of the previous experiments in this thesis. In Chapter 6.2 it can be seen that the InceptionV3 architecture outperformed the other networks on the fixed feature extraction approach. The advantage of training or fine-tuning CNNs is that the feature computation and classification of images in one step. The preprocessing and the feature computation and classification are described in the next paragraphs.

5.5.1 Preprocessing

For fine-tuning InceptionV3 the images are reshaped to 256x256 pixels. The intensity gets scaled to the range of $[0, 255]$. As described in the previous experiment, the single color channel gets tripled, to obtain an imitation of a RGB image. Then all images get normalized to zero mean and standard deviation 1.
For data augmentation, horizontal flip and random cropping an image of size 227x227 pixels out of the image of size 256x256 pixels was executed.

5.5.2 Feature Computation and Classification

To fine-tune a CNN, weights can be frozen, such that they stay unchanged during the new training process. The weights get frozen by layer. Also the number of layers that are fixed are adjustable.
Before starting the training, the last layer of the network needs to be adapted, so it outputs correct class probabilities.
In this thesis we tested two approaches on fine-tuning. One was to fine-tune all layers of InceptionV3 with a low learning rate. The other approach was to only load the weights

of the first 5 of 11 Inception modules and initialize the other weights randomly. This model was trained with a low learning rate as well.

CHAPTER 6

Results

In this chapter all results from the implemented approaches are documented and described. Different tests of each approach are compared. The discussion of the results is provided in chapter 7.

6.1 Feature Engineering

In Table 6.1 precision, f1-score, recall and AUC of the feature engineering approach are presented. These results were obtained by computing histograms of different features. The extracted features were the gradient magnitude, local binary pattern, histogram of oriented gradients, shape and curvature and the intensity, as described in Section 4.1. A comparison with related work can be found in Chapter 7.

F1-Score	**Precision**	**Recall**	**AUC**
$0,66$	$0,67$	$0,67$	$0,70$

Table 6.1: Feature Engineering results

6.2 Fixed Feature Extraction

For each of the CNN architectures we tested - AlexNet, VGG16, InceptionV3, and ResNet50 - we provide a table containing the results of TBC classification. Each experiment was tested on every model to ensure comparability. The influence of color normalization and setting class weigths were tested.

The results for AlexNet are in Table 6.2. The best result, highlighted in bold, was obtained without any preprocessing or augmentation. Color Normalization of the data

Data Augmentation	F1-Score	Precision	Recall
without color normalization	**0,73**	**0,73**	**0,73**
with color normalization	0,71	0,71	0,72
class weights	0,72	0,71	0,72

Table 6.2: AlexNet Fixed Feature Extraction results.

yielded 1-2% less precision, just like trying to balance the class weights in classification with the SVM.

Setting class weights for the training of the SVM did not make any difference on the classification with VGG16. Color normalization slightly worsened the results, as documented in Table 6.3.

Data Augmentation	F1-Score	Precision	Recall
without color normalization	**0,85**	**0,85**	**0,85**
with color normalization	0,84	0,85	0,84
class weights	**0,85**	**0,85**	**0,85**

Table 6.3: VGG16 Fixed Feature Extraction results.

With InceptionV3 as fixed feature extractor, the best results (see Table 6.4) were obtained using color normalization. This is because the network was pretrained on the ImageNet dataset using color normalization. Adding class weights had no influence on the performance. Without color normalization the networks precision is 3% worse than with color normalization.

Data Augmentation	F1-Score	Precision	Recall
without color normalization	0,83	0,83	0,83
with color normalization	**0,86**	**0,86**	**0,86**
class weights	**0,86**	**0,86**	**0,86**

Table 6.4: InceptionV3 Fixed Feature Extraction results.

ResNet50 was the last architecture that was evaluated and the results are in Table 6.5. With class weights, recall and f1-score, could be increased by 2%. Color normalization had nearly no influence on the result, precision is 1% better without color normalization. The other metrics stay unchanged.

Different results originate from the different architectures. In the different years of the ImageNet Large Scale Visual Recognition Challenge the presented networks were getting

Data Augmentation	F1-Score	Precision	Recall
without color normalization	0,72	0,75	0,73
with color normalization	0,72	0,74	0,73
class weights	**0,74**	**0,75**	**0,75**

Table 6.5: ResNet50 Fixed Feature Extraction results.

better year per year (see section 4.2.2). This trend is visible in our results, but for our data InceptionV3 gave the best results.

6.3 Fine-Tuning

Table 6.6 contains the results of the two main fine-tuning experiments performed in this thesis. For each dataset - training, validation and test - the F1-Score, Precision and Recall were calculated and additionally for the validation and test set the AUC is available. The results of the test set are highlighted in bold for each experiment.
As described in Section 5.5 InceptionV3 was fine-tuned. Fine-Tuning-11 refers to the approach on fine-tuning all 11 Inception modules of the network pretrained on ImageNet. For Fine-Tuning-5, only the weigths up to the fifth Inception module were kept while the other layers were randomly initialized and then the network was trained again. Both experiments were executed with the same learning rate.
As the related publications use the AUC as performance metric, we specify Fine-Tuning-5 as the better model, because it has the better AUC and compare it in Section 7.2.3 with related work.

Model	F1-Score	Precision	Recall	AUC
Fine-Tuning-11 training	0,75	0,77	0,74	-
Fine-Tuning-11 validation	0,51	0,62	0,44	0,75
Fine-Tuning-11 test	**0,63**	**0,73**	**0,55**	**0,78**
Fine-Tuning-5 training	0,78	0,78	0,81	-
Fine-Tuning-5 validation	0,66	0,64	0,68	0,81
Fine-Tuning-5 test	**0,66**	**0,67**	**0,65**	**0,79**

Table 6.6: InceptionV3 Fine-Tuning results.

The loss functions of the models are presented in Figure 6.1.The blue curve visualizes the training loss and the green curve the validation loss. Both loss functions show a promising curve, as the loss is always decreasing and does not start to increase, which indicates a well chosen learning rate. A low learning rate would be visible as linear

function, very high learning rates appear similar to exponential functions [LKJ].

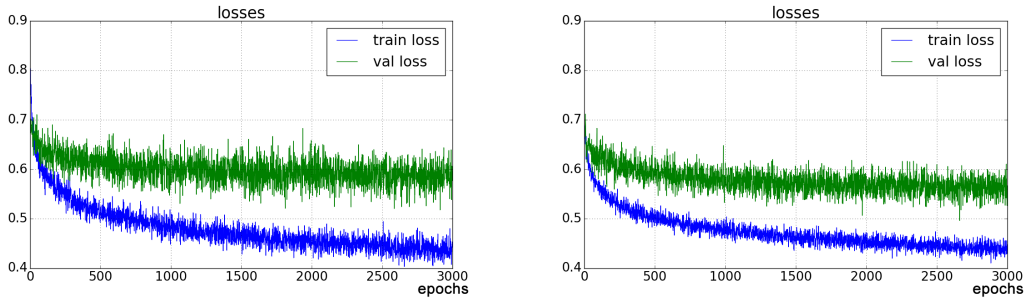

Figure 6.1: Plot of the loss function of Fine-Tuning-11 (left) and Fine-Tuning-5 (right). The loss function of the training set is visible in blue and the loss function of the validation set is visible in green.

In Figure 6.2 the AUC curve of the models over the epochs can be seen. The blue curve is the AUC of the validation set and the green curve the AUC of the test set. As the AUC curves in the right figure of the validation and test set are very similar it indicates that this model (Fine-Tuning-5) has a better generalization.

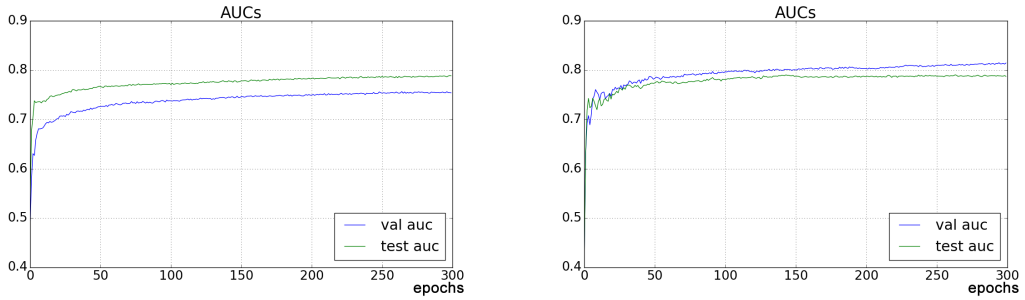

Figure 6.2: Plot of the AUC function of Fine-Tuning-11 (left) and Fine-Tuning-5 (right). The AUC of the validation set is visible in blue and the AUC of the test set is visible in green.

The advantage of Fine-Tuning-5 is that the later layers of the pretrained network with very specific features on natural RGB images do not have any influence as the weights are initialized randomly. The pretrained layers need more training and data to perform better on the TBC task compared to randomly initialized weights.

CHAPTER 7

Critical reflection

This chapter provides a critical reflection of the presented implementation and results. We summarize and compare the results of the different approaches implemented in this thesis. Then we compare the implementation and results with related publications in Section 7.2. In Section 7.3 open issues will be discussed.

7.1 Comparison of our Approaches

In Table 7.1 the results achieved in the different experiments of this thesis, are presented. The best result is highlighted in bold. Overall it can be observed that the best results were achieved using CNNs as fixed feature extractor and SVMs for classification.
One advantage of the fixed feature extraction approach is the classification with SVMs instead of the last fully-connected layers of CNNs. SVMs can outperform the classification part of CNNs. Huang et al. [HL06] showed that the combination of the advantages of SVMs and CNNs can result in better performance. With pixel-classification by SVMs they reached an error rate of 43,3%, with a CNN 7,2% and with the combination of both 5,9%. The performance gain results from the characteristics that CNNs are good at learning features, while SVMs are good in finding decision surfaces. This combination provided good performance for us too.
Another fact is that less data is needed to only train a SVM than to train or fine-tune a whole CNN. Still, we were able to benefit from well-trained feature extractors. So with little data, more effective classifications can be achieved with SVMs.
It has been shown before in multiple publications that the traditional approach of feature engineering can be outperformed with CNNs. For example Shin et al. [SRG+16] set the state-of-the-art on mediastinal lung nodule detection with fine-tuning CNNs. Lakhani et al. [LS17] also used fine-tuning to set the state-of-the-art in TBC classification.
This is because the features, which were selected to be extracted for classification with feature engineering, are focused on different types of appearances of TBC. However, this

does not mean that these extracted features are the best for classification. CNNs learn to extract the best features relevant to the task and then use these for classification. This is one main advantage of CNNs compared to traditional implementations.
In our approaches, we could not outperform the fixed feature extraction with fine-tuning, as stated by others, e.g. Shin et al. [SRG+16]. The reason for this is our dataset. We have a small dataset with heterogenous data containing different forms of TBC, like early stages of active TBC, which are hard to detect. With another network architecture, which is smaller than InceptionV3 and therefore needs less data to train, we probably could have obtained better results with the current dataset.
One key characteristic of CNNs is that they need long time to train, but after training, classification of a single image is achieved very fast, since the network already learned, what parts of the images are relevant for classification.

Model	**F1-Score**	**Precision**	**Recall**	**AUC**
Feature Engineering	0,66	0,67	0,67	0,70
AlexNet	0,73	0,73	0,73	0.71
VGG16	0,85	0,85	0,85	0,84
InceptionV3	**0,86**	**0,86**	**0,86**	**0,85**
ResNet50	0,74	0,75	0,75	0,73
Fine-Tuning-11	0,63	0,73	0,55	0,78
Fine-Tuning-5	0,66	0,67	0,65	0,79

Table 7.1: Summary of the best results of the different approaches used in this thesis.

7.2 Comparison with Related Work

In this section we try to draw a comparison with related publications. Therefore it is structured according to the different conducted experiments.

7.2.1 Feature Engineering

Our approach on feature engineering is based on the publication of Jaeger et al. [JKC+14]. Still we cannot draw a direct comparison with their approach, because we have a different dataset. Nevertheless we compare our results with the results of Jaeger et al. in the following paragraph.

The big difference in performance between this implementation and the implementation of Jaeger et al. results from the dataset. Jaeger et al. used publicly available datasets - Montgomery County chest X-ray set and Shenzen chest X-ray set. As stated by Jaeger et al. [JCA+14], these datasets only contain progressed stages of TBC, which are easily

detectable in radiographs. Our heterogenous dataset contains different types of TBC, hard detectable early stages of active TBC too. Jaeger et al. had less data samples (753) than we had in this thesis (1544). They showed the influence of different datasets on the performance in their publication. On the feature set A, which is most similar to our approach, a difference of up to 1.1% in AUC resulted from two different datasets.
Melendez et al. [MvGM+16] proposed a different approach using Multiple Instance Learning and Active Learning. As they state, the advantage of using Multiple Instance Learning is that the outlines of manifestations do not need to be found accurately as in traditional approaches.
The publication of Hogeweg et al. [HSM+15] is more focused on the generalization of different manifestations of TBC as described in section 2.1. They use different subsystems for different features and then combine them.
All the publications related to our approach show similar performance.

Publication	**AUC**
Feature Engineering	$0,70$
Jaeger et al. [JKC+14]	$0,87$
Melendez et al. [MvGM+16]	$0,88$
Hogeweg et al. [HSM+15]	$0,86$

Table 7.2: Summary of the best results of Feature Engineering of this and related work.

7.2.2 Fixed Feature Extraction

We could not find any related publications on the TBC classification task that used fixed feature extraction. The publication of Shin et al. [SRG+16], which covers the task of lung nodule detection, only states that fixed feature extraction yields worse results than fine-tuning CNNs. As seen in Table 7.1, we could not verify this statement. With more data we maybe could have gotten similar findings. With little data the approach of fixed feature extraction seems to be more effective.

7.2.3 Fine-Tuning

The state-of-the-art in TBC classification was set by Lakhani et al. [LS17] by fine-tuning AlexNet and GoogleNet pretrained on ImageNet and then combine the models for classification. They reached an AUC of 98% with each architecture and an AUC of 99% with an ensamble of GoogLeNet and AlexNet. Hwang et al. [HKJK16] published their results on fine-tuning AlexNet with an AUC of 96,7%.
The main advantage of Hwang et al. compared to the presented work in this thesis and the publication of Lakhani et al. is the size of their dataset. With 12.648 images their dataset was much bigger than the dataset of Lakhani et al. with 1007 images and the dataset used in this thesis with 1544 images. Especially when training CNNs this can be a big gain. This amount of data is the main reason for a better performance in their

publication compared to the presented approach.
Another reason could be that they used input data with a higher resolution of 500x500 pixels compared to the approach in this thesis and of Lakhani et al. [LS17] with 227x227 pixels. They used the same data augmentation as in this thesis, random cropping and horizontal mirroring.
Despite the small dataset, Lakhani et al. [LS17] were able to outperform the publication of Hwang et al. [HKJK16]. They used more aggresive data augmentation for performance gain and they put emphasis on a balanced dataset with 492 tuberculotic and 515 healthy images. Additionally to random cropping and mirroring they used rotations of 90°, 180° and 270° and contrast limited adaptive histogram equalization. Their approach with fine-Tuning AlexNet outperformed Hwang et al. with a gain of 1,3% in AUC.
One advantage in training GoogLeNet compared to AlexNet, especially when only a small dataset is available, is the smaller number of trainable parameters. This does not seem to make any difference in the approach of Lakhani et al., as they could reach similar AUC on both architectures.
It is hard to compare the performance of CNNs, as it mainly depends on the dataset used for training and testing. Lakhani et al. use publicly available datasets, which where reviewed multiple times. These public datasets therefore contain less heterogenous data, consisting of radiographs with progressed stages of TBC, which are easier to detect [JCA+14]. The dataset in this work directly comes from a hospital. It is a heterogenous dataset, which contains different types of TBC. It contains early stages of TBC as well, which are hard to detect using radiographs only.

Publication	**AUC**
Fine-Tuning-5	0,79
AlexNet Hwang et al. [HKJK16]	0,967
AlexNet Lakhani et al. [LS17]	0,98
GoogLeNet Lakhani et al. [LS17]	0,98
Ensemble Lakhani et al. [LS17]	0,99

Table 7.3: Summary of performance of fine-tuning a CNN of this and related work.

7.3 Discussion of Open Issues

The main unresolved issue in the TBC classification with CNNs is the data acquisition. As CNNs depend on the dataset, it is important to have a big dataset, which is as diverse as possible and compatible with the given task. A bigger dataset is necessary for generalization and performance gain. Additionally, big, publicly available, datasets are important to objectively compare different approaches.
Especially for a small dataset - like the one used in this thesis - it could be benficial to use a network with a small number of parameters for fine-tuning. For example GoogLeNet

with 5 million parameters could yield better performance, as overfitting should be easier to prevent.
Lakhani et al. [LS17] achieved a performance gain by combining different good working models for the task. The combination of different models is an open issue of this thesis.
Applying pretrained CNNs as fixed feature extractor could be interesting when extracting features at different layers. As it has been mentioned in this thesis more general features are found in earlier layers. Therefore it could be useful to extract more general features at earlier layers and classify them afterwards. Testing the features of different layers could also provide information for fine-tuning, as one can find out at which layer the features start to get specific for the pretrained task. Additionally compositions of feature extraction models, which extract different features at a different layer could be worth a try.
Another open issue is to adapt the SVM, such that the gradients can be backpropagated and lower level features can be learned as described by Tang et al. [Tan13].

List of Figures

List of Tables

Bibliography

[AGM98] S. G. Armato, M. L. Giger, and H. MacMahon. Automated lung segmentation in digitized posteroanterior chest radiographs. *Academic Radiology*, 5(4):245–255, 1998.

[Bur98] Christopher J C Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[Cho15] François Chollet. Keras Documentation, 2015. `https://keras.io`, accessed 23.11.2017.

[Coh] Joseph Paul Cohen. *Visualizing CNN architectures side by side with mxnet.* `http://josephpcohen.com/w/visualizing-cnn-architectures-side-by-side-with-mxnet/`, accessed 23.11.2017.

[Dia17] Diagnostic Imaging Dataset. Diagnostic Imaging Dataset Annual Statistical. Technical report, 2017. `http://www.wjgnet.com/1949-8470/full/v6/i1/1.htm`, accessed 23.11.2017.

[DPU+09] D. W. De Boo, M. Prokop, M. Uffmann, B. van Ginneken, and C. M. Schaefer-Prokop. Computer-aided detection (CAD) of lung nodules and small tumours on chest radiographs. *European Journal of Radiology*, 72(2):218–225, 2009.

[EEZ+06] M Everingham, M Everingham, A Zisserman, A Zisserman, C Williams, and C Williams. The PASCAL visual object classes challenge 2006 (VOC2006) results. *Workshop in ECCV06, May. Graz, Austria*, 2006, 2006.

[Fel98] Christiane Fellbaum, editor. *WordNet: an electronic lexical database*. MIT Press, 1998.

[FNVV98] Alejandro Frangi, Wiro Niessen, Koen Vincken, and Max Viergever. Multiscale vessel enhancement filtering Medical Image Computing and Computer-Assisted Interventation — MICCAI'98. In *Medical Image Computing and Computer-Assisted Interventation — MICCAI'98*, volume 1496, pages 130–137. 1998.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning–book. *MIT Press*, 521(7553):800, 2016.

[HKJK16] Sangheum Hwang, Hyo-Eun Kim, Jihoon Jeong, and Hee-Jin Kim. A novel approach for tuberculosis screening based on deep convolutional neural networks. In *SPIE Medical Imaging*, page 97852W, 2016.

[HL06] Fu Jie Huang and Yann LeCun. Large-scale learning with SVM and convolutional nets for generic object categorization. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 284–291, 2006.

[HSM+15] Laurens Hogeweg, Clara I. Sánchez, Pragnya Maduskar, Rick Philipsen, Alistair Story, Rodney Dawson, Grant Theron, Keertan Dheda, Liesbeth Peters-Bax, and Bram Van Ginneken. Automatic Detection of Tuberculosis in Chest Radiographs Using a Combination of Textural, Focal, and Shape Abnormality Analysis. *IEEE Transactions on Medical Imaging*, 34(12):2429–2442, 2015.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[JCA+14] Stefan Jaeger, Sema Candemir, Sameer Antani, Yi-Xiang J Wang, Pu-Xuan Lu, and George Thoma. Two public chest X-ray datasets for computer-aided screening of pulmonary diseases. *Quantitative imaging in medicine and surgery*, 4(6):475–477, 2014.

[JKC+14] Stefan Jaeger, Alexandros Karargyris, Sema Candemir, Les Folio, Jenifer Siegelman, Fiona Callaghan, Zhiyun Xue, Kannappan Palaniappan, Rahul K. Singh, Sameer Antani, George Thoma, Yi Xiang Wang, Pu Xuan Lu, and Clement J. McDonald. Automatic tuberculosis screening using chest radiographs. *IEEE Transactions on Medical Imaging*, 33(2):233–245, 2014.

[JOP+] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. Online; accessed 27.08.2017.

[JWS+09] 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.

[KSG12] Alex Krizhevsky, Ilya Sutskever, and Hinton Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems 25 (NIPS2012)*, pages 1–9, 2012.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323, 1998.

[Leu99] A N Leung. Pulmonary tuberculosis: the essentials. *Radiology*, 210(2):307–322, 1999.

[LKJ] Fei-Fei Li, Andrej Karpathy, and Justin Johnson. CS231n Convolutional Neural Networks for Visual Recognition. `http://cs231n.github.io/`, accessed 23.11.2017.

[LS17] Paras Lakhani and Baskaran Sundaram. Deep Learning at Chest Radiography: Automated Classification of Pulmonary Tuberculosis by Using Convolutional Neural Networks. *Radiology*, 284(2):574–582, 2017.

[Mas11] D Mason. Pydicom: An Open Source DICOM Library. *Medical Physics*, 38(6):3493, 2011.

[McC86] R K McConnell. Method of and apparatus for pattern recognition, 1986. `http://www.google.co.uk/patents/US4567610`, accessed 23.11.2017.

[MEM02] Peter Mildenberger, Marco Eichelberg, and Eric Martin. Introduction to the DICOM standard. *European Radiology*, 12(4):920–927, 2002.

[MMA+13] P Maduskar, M Muyoyeta, H Ayles, L Hogeweg, L Peters-Bax, and B van Ginneken. Detection of tuberculosis using digital chest radiography: automated reading vs. interpretation by clinical officers. *The international journal of tuberculosis and lung disease : the official journal of the International Union against Tuberculosis and Lung Disease*, 17(12):1613–1620, 2013.

[MvGM+16] Jaime Melendez, Bram van Ginneken, Pragnya Maduskar, Rick H. H. M. Philipsen, Helen Ayles, and Clara I. Sanchez. On Combining Multiple-Instance Learning and Active Learning for Computer-Aided Detection of Tuberculosis. *IEEE Transactions on Medical Imaging*, 35(4):1013–1024, 2016.

[Oli07] Travis E. Oliphant. Python for scientific computing. *Computing in Science and Engineering*, 9(3):10–20, 2007.

[OPH94] T Ojala, M Pietikainen, and D Harwood. Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. In *Proceedings of 12th International Conference on Pattern Recognition*, pages 582–585 vol.1, 1994.

[PV11] Fabian Pedregosa and G Varoquaux. *Scikit-learn: Machine learning in Python*, volume 12. 2011. http://dl.acm.org/citation.cfm?id=2078195, accessed 23.11.2017.

[RASC14] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 512–519, 2014.

[SEZ+13] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *arXiv preprint arXiv*, page 1312.6229, 2013.

[Shl] John Shlens. Research Blog: Train your own image classifier with Inception in TensorFlow. https://research.googleblog.com/2016/03/train-your-own-image-classifier-with.html, accessed 23.11.2017.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June, pages 1–9, 2015.

[SRG+16] 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.

[SVI+15] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. 2015.

[SZ15] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICRL)*, pages 1–14, 2015.

[Tan13] Yichuan Tang. Deep Learning using Linear Support Vector Machines. *Deeplearning.Net*, 2013. http://deeplearning.net/wp-content/uploads/2013/03/dlsvm.pdf, accessed 23.11.2017.

[The16] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, page 19, 2016.

[Uni16] United Nations. The Sustainable Development Goals Report. pages 1–56, 2016. http://www.un.org/sustainabledevelopment/sustainable-development-goals/, accessed 23.11.2017.

[VCV11] Stéfan Van Der Walt, S. Chris Colbert, and Gaël Varoquaux. The NumPy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, 13(2):22–30, 2011.

[vGSJC15] 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, 2015.

[vGSL06] Bram van Ginneken, Mikkel B. Stegmann, and Marco Loog. Segmentation of anatomical structures in chest radiographs using supervised methods: A comparative study on a public database. *Medical Image Analysis*, 10(1):19–40, 2006.

[WHO16] WHO. WHO Global tuberculosis report 2016. Technical report, 2016. http://www.who.int/tb/publications/global{_}report/en/, accessed 23.11.2017.

[YCBL14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, abs/1411.1:3320—-3328, 2014.