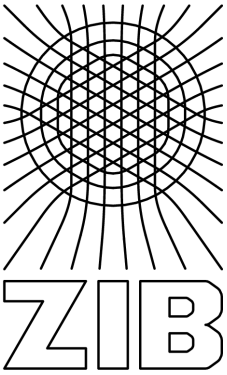
INSTITUT FÜR
MATHEMATIK UND
INFORMATIK
DER FREIEN UNIVERSITÄT BERLIN

AG THERAPY
PLANNING
DES ZUSE INSTITUTS BERLIN



Bachelor Thesis

Comparison of 2D and 3D CNNs for Classification of Knee MRI

Mona Prendke

First Supervisor:	Prof. Dr. Tim Conrad
Second Supervisor:	Dr. Stefan Zachow
Semester:	Winter Term 2018/19
Author:	Mona Prendke
Matriculation-Nr.:	- deleted due to reasons of privacy -
Address:	- deleted due to reasons of privacy -
Email:	mona.prendke@fu-berlin.de
Subject:	Bachelor Bioinformatics
Submission:	04 February 2019

Abstract

Osteoarthritis (OA) is a degenerative joint disease with increasing prevalence that involves pain symptoms, reduction of functionality, and constitutes one of the most common causes of disability in adults. It occurs most often in the knee joint. Since there exists no cure and only insufficient diagnostic methods, hopes for improvement are based on automated approaches. In this context, the Osteoarthritis Initiative collected data on the progress of OA including many images from magnetic resonance tomography (MRI). This MRI data was used by the Zuse Institute Berlin to develop a fully automated segmentation method for knee bone and cartilage to better extract biomarkers from the images as a starting point to automated diagnosis and further research on OA. In this thesis, I present two convolutional neural networks (CNNs) for classification of the input MRI scan's laterality with 100% accuracy on the testing set, which can be included in the segmentation pipeline to make it more efficient, faster, and less data dependent. Using spatial data with high resolution like MRI scans as input to CNNs poses certain problems since the convolutions of most ready-to-use CNNs work on two-dimensional grids, three-dimensional convolution operations are computationally more expensive, and most ready-to-use CNNs work with input images of lower resolution. In this thesis, I explored, compared, and evaluated three different approaches that meet this challenge. To achieve this, I selected one pretrained CNN implementation from each of the solution approach categories (*i*) *2D CNN with manual slice selection*, (*ii*) *multi view CNN*, and (*iii*) *3D CNN*, prepared the raw MRI scans featuring one knee joint to be suitable inputs for each CNN, fine-tuned each CNN on the MRI input images to solve the task of classifying their laterality, and evaluated their performance. Then I compared the three approaches and their representation implementations in terms of network architecture, training progress, and test performance and discussed their suitability for the desired task. I conclude that the less complicated models, *2D CNN with manual slice selection* and *3D CNN*, are both well-suited to solve the task with perfect accuracy. *2D CNN with one slice as input* requires less data to converge, while *3D CNN* has the advantage of not needing manual slice selection.

Summary

The subject of this thesis is comparing 2D and 3D CNNs for classification of laterality in knee joint MRI scans. Its main accomplishment is the presentation of two CNNs that solve this task with 100% accuracy. Steps to achieve this:

- Literature research on 2D and 3D CNNs, medical image analysis, machine learning in medicine, anatomy of the knee joint, and osteoarthritis.
- Manual exemplary visual analysis of knee MRI scans identifying the visible range and laterality-specific anatomical features.
- Categorization of solution approaches to handling 3D data with CNNs into three categories: 2D CNN with manual slice extraction, multi view 2D CNN, and 3D CNN.
- Selection of one CNN implementation per category for analysis: VGG19, AlexNet multi view, and VoxNet.
- Development of tools for data partitioning into training, validation, and test set, image loading, and preprocessing, i.e. normalization, downsampling, change in medical orientation plane, and slice extraction.
- Grid search for preselection of slice input for VGG19.
- For each of the three CNN implementations: preprocessing, training, validation of training progress, evaluation, and analysis of results.
- Comparison of the three CNNs in regard to test set accuracy, training and validation accuracy, convergence, and duration of preprocessing, training, and evaluation.
- Assessment of the three CNNs' suitability for utilization in ZIB knee segmentation pipeline and other medical applications.
- Classification of results in a broader context and outlook to future possibilities.

Acknowledgment

The work of this thesis was completed in cooperation with the research group for Therapy Planning at the Zuse Institute Berlin. I would like to thank them for making this possible and for providing me with all the necessary facilities and data for the computing experiments.

My sincere thanks to Prof. Dr. Tim Conrad from Freie Universität Berlin and Dr. Stefan Zachow from Zuse Institute Berlin for their valuable academic support and helpful advice.

I take this opportunity to express my special gratitude to Felix Ambellan and Alexander Tack, doctoral students of the aforementioned research group, for their guidance, constructive discussions and motivation throughout the whole research and writing process.

Last but not least, I would like to thank my partner for his patience, reassurances, and proofreading, as well as my parents for their unceasing support and encouragement.

Contents

1 Introduction

Imaging methods in medical research as well as in everyday medicine produce image data that is traditionally analyzed manually by trained specialists, such as radiologists. Alternative methods were introduced by the application of computer-aided image analysis with at first rather fruitless attempts as early as in the 1960s (Meyers et al. 1964; Doi 2007; Engle 1992). Machine learning (ML) approaches on expert problems in various fields multiplied and became increasingly successful in the 1980s. This, along with the increasing amount of available data, built the foundation for the progress in computer vision at that time and of it growing ever since (Freeman et al. 2008; Sebag 2014). In the 1990s, computer scientists began to apply the mathematical methods that had proven successful in solving non-medical image analysis problems to medical images, thereby founding the field of medical image analysis (MIA), a subfield of computer vision, as we understand it today (Wells 2016).

Automated approaches to MIA help improve our understanding and treatment of diseases in many ways. Computer-aided diagnosis (CAD) can help radiologists find a better diagnosis (Doi 2007). For example, the CAD of vertebral fractures on lateral chest radiographs, which are missed in more than 30% to 50% by medical personnel, can help recognize osteoporosis early (Doi 2007; Kasai et al. 2006). Similarly, pulmonary nodules in lung computer tomography (CT) scans can be classified into different levels of solidity, helping to categorize them as benign or malignant as early as possible to hopefully decrease lung cancer mortality (Jacobs et al. 2014). Computer-based image analysis is also a prerequisite for most computer-assisted surgery tasks such as implant placement (Bover-Ramos et al. 2018; Hernandez et al. 2017), as well as it can be a part of surgical planning e.g. in rhinoplasty (Singh & Pearlman 2017). As can be seen by this wide range of examples, MIA has diverse application possibilities in therapy planning.

MIA, as described, above employs a variety of different techniques, such as the morphing of images (Singh & Pearlman 2017), enhancement filtering or segmentation followed by feature extraction and classification of feature input (Jacobs et al. 2014; Kasai et al. 2006). Especially the latter method was state of the art and used frequently in MIA before the triumphant advance of deep learning including convolutional neural networks (CNNs) (LeCun et al. 1990; Suzuki 2017). CNNs have been widely popular and well-used to

process and analyze complex pictures for computer-based image analysis in general (Suzuki 2017). This trend started in 2012 with a CNN-based method, AlexNet (Krizhevsky et al. 2012), overwhelmingly winning at the prestigious 'ImageNet Large Scale Visual Recognition Competition' (ILSVRC) (Russakovsky et al. 2015). Since then, their use has successfully been explored in many other applications, for instance in the game of Go (Silver et al. 2016; Google Deepmind 2019), in video analysis (Karpathy et al. 2014; Wang et al. 2018), and drug discovery (Wallach et al. 2015).

1.1 Background and Motivation

'Osteoarthritis (OA) is a long-term chronic disease characterized by the deterioration of cartilage in joints which results in bones rubbing together and creating stiffness, pain, and impaired movement.' (World Health Organization 2012) OA 'accounts for more disability among the elderly than any other disease' (European Musculoskeletal Conditions Surveillance and Information Network 2012). Due to impairment of functionality and pain in the patient 'it places a major burden on individuals, communities, health systems, and social care systems.' (Tanna 2013). OA prevalence in developed countries is mostly around 10% to 20% in adults (Fuchs et al. 2017; Helmick et al. 2008; European Musculoskeletal Conditions Surveillance and Information Network 2012) and will rise further in the future due to the population growing and aging as well as to risk factors such as obesity increasing. At the moment, no cure exists and treatment is mostly palliative. Therefore, research on understanding, diagnosing and treating OA will become even more important in the future. At present, OA is diagnosed by physical examination and manual radiographic detection of pathological changes such as 'loss of joint space, subchondral sclerosis, cysts and osteophytes' (European Musculoskeletal Conditions Surveillance and Information Network 2012). However, 'these diagnostic tools have low sensitivity and specifity' (World Health Organization 2012). Automated approaches to diagnosis and research of OA are hoped to improve this situation in the future.

The research group for Therapy Planning headed by Dr. Stefan Zachow at the Department of Visual Data Analysis is a part of the Division for Mathematics for Life and Materials Sciences at the Zuse Institute Berlin (ZIB) who aim at 'identify[ing] needs and possibilities for computer assisted, model-guided treatment planning in close collaboration with domain experts in view of improved therapeutic results, and [strive to] implement interactive and intuitive, graphics-based software prototypes for decision support and therapy planning in order to demonstrate clinical applicability' (Research Group Therapy Planning at Zuse

Institute Berlin 2019).

In this context, they developed a 'fully automated segmentation method for knee bone and cartilage [from magnet resonance imaging (MRI)] by combining the advantages of Statistical Shape Model-based regularization with CNN-based classification of voxel intensities' (Ambellan et al. 2019). Such 'precise segmentations are [...] a prerequisite for computer-based surgical planning of interventions affecting the knee' (Ambellan et al. 2019). To use biomarkers found in radiographic images for a computer-based diagnosis of OA of the knee, segmentation is a necessary prestep. Since 'manual segmentation is tedious, time-consuming, and labour intensive', automated approaches need to be developed (Tack et al. 2018). This development is very data dependent because it needs manually segmented MRI scans for construction and training.

Structures to be found by segmentation of radiographic images that feature exactly one left or right knee are obviously vaguely symmetric to the median plane, but asymmetric within one knee. Therefore, the general form of segments differs depending on laterality. With this in view, it is desirable to know before segmentation if the input is of a left or a right knee.

Hence, a method distinguishing left from right knees, if accurate enough, could be included in the segmentation pipeline to hopefully make the algorithm more efficient, faster, and less data dependent due to more available data with labels on MRI laterality than segmentation. In this connection, my task was researching methods to classify the laterality of knee joint MRI scans.

1.2 General Solution Approach

At first sight, the knee joint appears to be vaguely symmetric to its middle sagittal plane. However, at closer examination, certain anatomical structures in one knee joint are visibly asymmetric to this plane, and can therefore be used to infer laterality. As shown in Fig. 1.1, the knee is composited of the tibiofemoral joint that articulates thigh bone (femur) and shin bone (tibia), and the patellofemoral joint that articulates the kneecap (patella) and the thigh bone. In the tibiofemoral joint, the femoral medial and lateral condyles are biconvex, and the tibial plateaus are slightly concave. This shape incongruence is compensated by two crescent-shaped menisci in-between the femoral condyles and the tibial plateaus. The medial part of the distal femur epiphysis with the medial condyle is bigger and located more distal than the lateral epiphysis with the lateral condyle, presenting a reference point for deducing laterality. Another femur structure that can be used for this is the sulcus

popliteus on the lateral epiphysis, the groove where the popliteus muscle originates (Zilles & Tillmann 2010, pp. 255-256). Additionally, the posterior patella is asymmetrical with a small, convex medial facies and a bigger, concave lateral facies (Zilles & Tillmann 2010, p. 245).

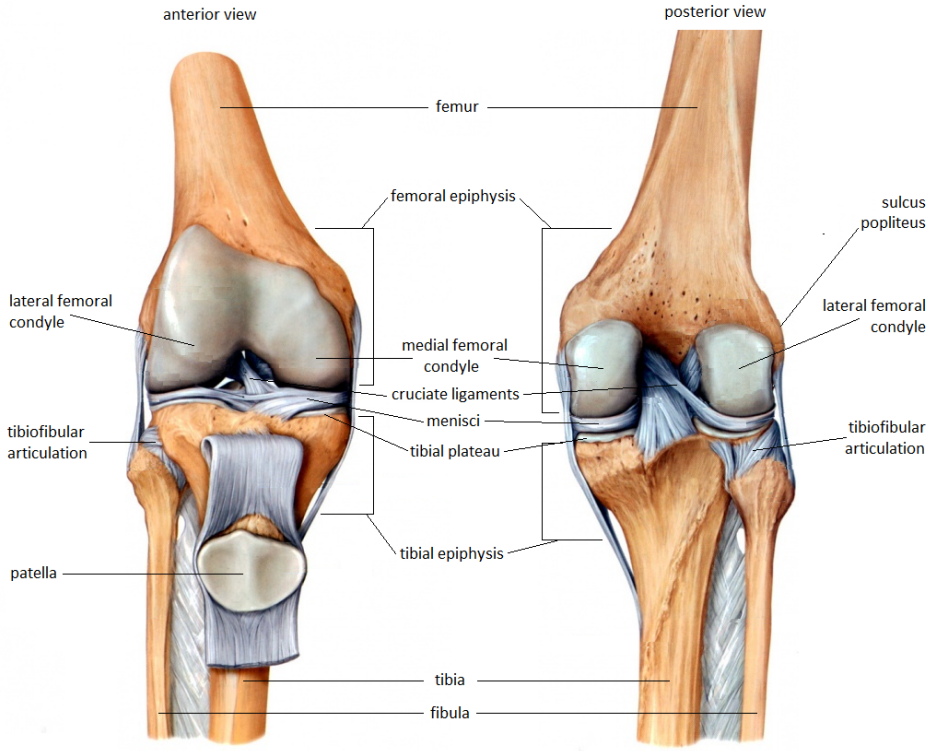
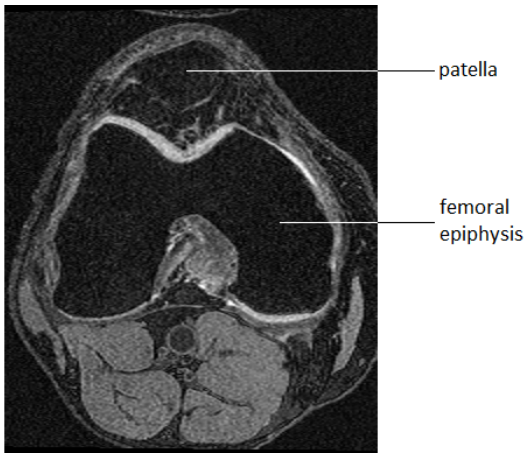
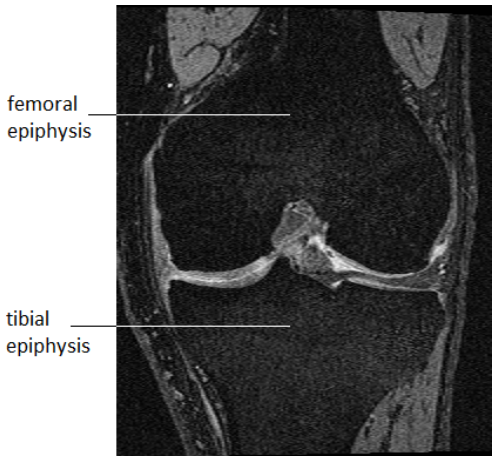


Figure 1.1: Anatomical knee structure (Muhic 2016).

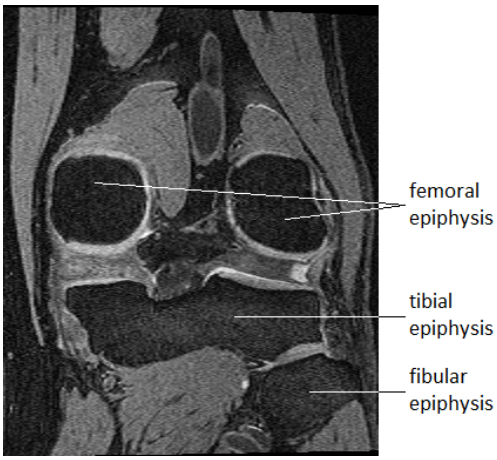
The tibiofibular articulation that connects tibia and calf bone (fibula) is also shown in a knee MRI scan due to its spatial proximity, though not technically a part of the knee joint. This articulation is on the lateral side of the tibia and therefore gives another hint for distinguishing a left from a right knee (Zilles & Tillmann 2010, p. 246). Furthermore, the cruciate and collateral ligaments of the tibiofemoral joint, which have a stabilizing and kinetic function, are asymmetrical, too (Zilles & Tillmann 2010, p. 257). As can be seen by all these laterality-specific anatomical structures, classifying the three-dimensional image of a knee as left or right is a well-defined problem. Moreover, these anatomical structures can be seen in MRI scans of the knee (Fig. 1.2). Generally speaking, an appropriate method should therefore be able to learn decision making on laterality from the input knee joint MRI scans.



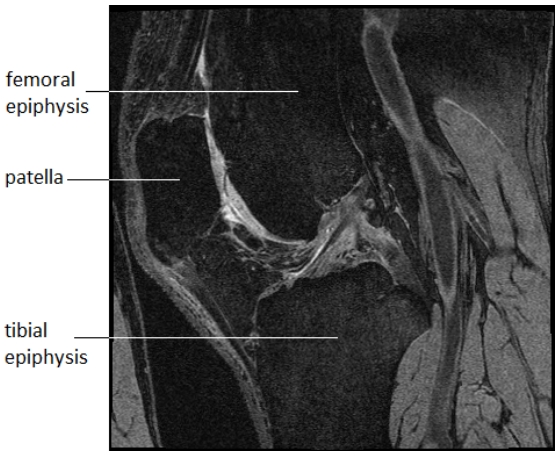
(a) axial-193



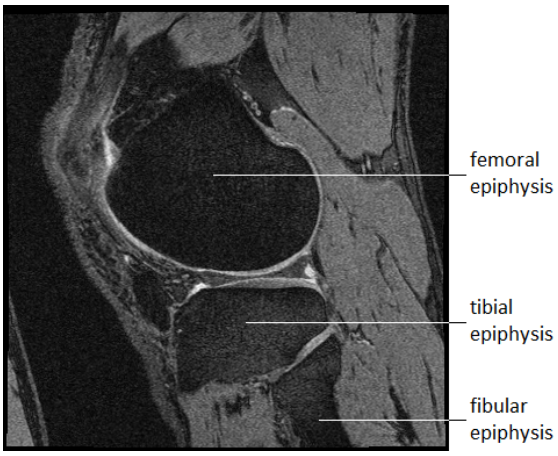
(b) coronal-194



(c) coronal-258



(d) sagittal-81



(e) sagittal-120

Figure 1.2: Exemplary slices from a raw MRI scan of resolution dimensions (sagittal-160, axial-384, coronal-384) that illustrate the laterality-specific anatomical structures.

MRI measures the spin relaxation time of excited atoms '(most notably hydrogen which is abundant in the human body), which varies with tissue type, molecular composition and functional status.' (Toennies 2012, p. 23) These distribution measurements are then projected onto multiple, ordered, two-dimensional slices through the human body, forming a cubical voxel grid that represents the volume (Toennies 2012, pp. 44-51). In analyzing MRI scans, one of the main challenges is handling the three-dimensionality, because it is a unique feature in medical 3D scans. Medical as well as everyday 2D images are obviously distinct from medical 3D scans due to their lack of a third dimension. Moreover, 3D representations of everyday objects differ from medical 3D images in terms of the distribution of their information content. The main information of an object in an ordinary 3D image is its surface and shape, whereas in medical 3D images, it is tissue distribution.

To classify the MRI scans, I had to research methods that met the challenge of handling this kind of three-dimensionality. In order to find a technical solution to this problem, in this thesis, I compare CNNs with different approaches to handling spatial data.

CNNs exploit the grid-like structure of their input data. A CNN for classification is usually comprised of several convolutional layers followed by at least one fully connected layer forming the classification part of the network (see Fig. 1.3). Each convolutional layer typically contains three stages: First, a convolution stage producing a set of linear activations by performing multiple convolutions in parallel. Second, a detector stage where a nonlinear activation function is applied to each linear activation e.g. rectified linear activation function (ReLU). Third, a pooling stage that 'replaces the output of the net at a certain location with a summary statistic of the nearby output' (Goodfellow et al. 2016, p. 330).

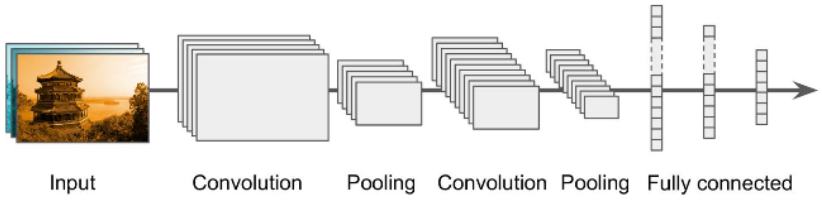


Figure 1.3: The structure of a typical convolutional neural network (Geron 2017, p. 327, Fig. 13.9).

CNNs use sparse interactions: The sliding window (and the weight matrix/kernel) in the convolutional layers is smaller than the input and deployed only locally. The weights in the convolutional layers are tied, meaning that the same kernels are used on every location in the picture. By this parameter sharing, the same features are being extracted on different

parts of the picture, facilitating an equivariance of the data to translation (Goodfellow et al. 2016, p. 332).

1.3 Literature Review of Related Work

As an initial point of my work, I caught up on the articles already gathered by the ZIB Therapy Planning Research Group on 2D and 3D CNNs (Çiçek et al. 2016; Dubost et al. 2017; Qi et al. 2016). I also performed searches on PubMed (US National Library of Medicine at the National Institutes of Health 2019), Google Scholar (Google Scholar 2019), Arxive (Cornell University 2019) and the online library catalogues of the Freie Universität (Universitätsbibliothek der Freien Universität Berlin 2019), Technische Universität (Universitätsbibliothek der Technischen Universität Berlin 2019) and Humboldt Universität Berlin (Universitätsbibliothek der Humboldt Universität Berlin 2019) with combinations of different search terms such as CNN, multi view, 3D, 2D, review, and MRI already filtering for publishment within the last ten years. In doing so, I found about 500 papers. Then I filtered these papers in two steps by title and then abstract, extracting 25 papers that seemed suitable for my approach. Additionally, I followed articles' references that sounded promising, thereby accumulating more sources.

Most CNNs partaking at ILSVRC and other computer vision competitions are readily available with pretrained weights, eg. AlexNet (Krizhevsky et al. 2012), VGG16, VGG19 (Simonyan & Zisserman 2014), ResNet (He et al. 2016), and Inception (Szegedy et al. 2015). Using spatial data like MRI or CT scans as input to CNNs poses certain problems since the convolutions of most ready-to-use CNNs work on two-dimensional grids and three-dimensional convolution operations are computationally more expensive (Qi et al. 2016). The simplest approach is to use one 2D projection (one slice) of the 3D image as input to a simple 2D CNN (*S2DCNN*). Current CNNs for 3D image data that actually take up the challenge of a three-dimensional input can be categorized into two widespread approaches: multi view CNNs (*MVCNNs*) and three-dimensional CNNs (*3DCNNs*).

In a *MVCNN* approach, at first, multiple views of one 3D image, such as recordings from a perspective rotation around the image's object, are processed independently. In a second step, the outcomes are combined to result in a classification or segmentation of the input image. The first widely cited publication using a *MVCNN* successfully was Su et al. (2015). Since then, this approach has also proven to be suitable for MIA, among others for segmentation of left atrium and proximal pulmonary veins (Mortazi et al. 2017), as well as estimation of ventricular volume (Luo et al. 2018), both on cardiac MRI scans.

The concept of a *3DCNN* approach is using three-dimensional convolutions. Initially, input images to CNNs with more than two dimensions were often depth images originating from RGB-D cameras and commodity range sensors (Gupta et al. 2014; Qi et al. 2016). In contrast to MRI, raw RGB-D images have only up to three additional values. Therefore, they have less complexity in the third dimension and are usually labeled as 2.5D. Since *3DCNNs* are often used for object recognition in robotics, object-detection should be independent from the viewing angle. For this reason, subsequent CNN methods for more than two-dimensional images usually reconstructed volumetric voxel grids from the RGB-D images for input (Wu et al. 2015; Maturana & Scherer 2015). Numerous *3DCNN*-based approaches have been implemented to solve MIA problems, including classification of Alzheimer's disease (Basaia et al. 2018; Khvostikov et al. 2018), tumor segmentation (Chen et al. 2018), liver segmentation (Çiçek et al. 2016) as well as brain lesion (Dubost et al. 2017) and pulmonary nodule detection (Zhu et al. 2017).

When working with either of the two approaches, 'there is an inherent trade-off between increasing the amount of explicit depth information (3D models [*3DCNN*]) and increasing spatial resolution (projected 2D models [*MVCNN*])' (Su et al. 2015). One general advantage of using *MVCNNs* when training CNNs from scratch is the vastly available amount of image data that can be used for learning low-level features in pre-training (Su et al. 2015). Qi et al. (2016) noticed and investigated a performance gap with *MVCNNs* outperforming *3DCNNs* in literature. They succeeded in replicating the result and suggested that different input resolution is probably not the only reason, but also *3DCNNs* being relatively new and their architectures not yet as sophisticated as two-dimensional CNNs.

Tajbakhsh et al. (2016) showed that it is usually not necessary to train a CNN from scratch on medical image data if pre-trained weights for the desired network architecture exist. On the contrary, using a pre-trained CNN with adequate fine-tuning in many cases even outperformed a CNN trained from scratch. Additionally, 'fine-tuned CNNs were more robust to the size of training sets CNNs than trained from scratch' (Tajbakhsh et al. 2016).

1.4 Task

In my thesis, I explore, compare and evaluate the different categories of handling spatial input data carved out above. To achieve this, I selected one well-established, pre-trained CNN implementation with good documentation from each of the categories *S2DCNN*, *MVCNN* and *3DCNN* to represent the respective approach. I prepared the raw MRI scans of knee articulations to be suitable inputs for each CNN, fine-tuned each CNN on the

preprocessed MRI scans to solve the task of classifying their laterality, and evaluated their performance. Then I compare the three approaches and their selected implementations in terms of network architecture, training progress, and test performance and discuss their suitability for the desired task. Finally, I generalize my findings and give an outlook. This thesis assumes that the reader has a basic knowledge of ML, especially of deep neural networks.

2 Data and Implementation Prerequisites

I had access to 9494 MRI scans obtained from an already existing local repository of the Osteoarthritis Initiative (OAI) (Osteoarthritis Initiative 2019) database at the ZIB. My general approach was using published, pretrained CNNs that had proven successful at a similar task. Then, broadly speaking, I finetuned the network weights in the fully connected as well as in some convolutional layers. For the finetuning process, I used 70% of the MRI scans as training set and 20% as validation set. The remaining 10% of the images were used for evaluation as a holdout data set. This split has proven to be effective in general for big amounts of data (Amazon 2019; Google Machine Learning 2019; Olivier Moindrot 2018).

2.1 Analyzed MRI Scans

The OAI is a longitudinal cohort study documenting the natural history of knee OA incidence and progression from 2003 until 2014 and providing public access to clinical and radiographic image data of their participants. The study recruited approximately 4800 people with, or at high risk of, symptomatic femoral-tibial knee OA at four clinical centers in the USA. The participants had various ethnic backgrounds, were aged 47-79 years, and 58% of them were female (Eckstein et al. 2014). The MRI scans were produced by using standardized 3 Tesla MRI (3T-MRI) scanners and protocols. The imaging technique used for recording the scans used in this thesis was 3D double-echo steady-state (DESS) sagittal with selected water excitation with 0.7 mm slice thickness and 0.37 mm $\times$ 0.37 mm in-plane area (Eckstein et al. 2007). When the ZIB obtained the data from the OAI, the data were already labeled by the radiologist who performed the MRI. Of the 9494 given images 4735 featured left and 4759 featured right knees. All but five MRI scans of left and four MRI scans of right knees had exactly 160 slices. I excluded those which differed from 160 slices due to concerns of corrupt data. Each slice had a resolution of (384×384) voxels.

2.2 Technical Prerequisites

The selection of tools and models was generally based on the preference for open source software and aspects of consistency across all approaches to maximize comparability of

results and enable reusability of implementation. All new code was written in Python. TensorFlow and CUDA made the code run on GPU. The GPU used for measurements of preprocessing and training time was an NVIDIA GeForce GTX 1080 Ti. In this context, eight Intel Xeon X5550 CPUs were utilized. Other experiments also used a ZIB GPU cluster with four NVIDIA Tesla P 100.

2.3 Image Loading and Preprocessing Prerequisites

Training the models required the development of tools to load, prepare and deliver the MRI scans to each network. For representation of image data, I used Numpy arrays and Pandas data frames. Using pretrained models with a generally fixed architecture meant that the model could not be easily changed to conform with the raw MRI data format. Instead, the MRI scans had to be preprocessed to fit into the input layer of each network. To that end, I used various Python site packages: Dicom was used for loading the original image. Open-cv/Cv2 provides functions to squeeze and resize the image to the desired dimensions and to project it from grayscale to RGB if needed. A Numpy function normalized the images. More details will be given at each solution approach's chapter.

The training data set of 6639 MRI scans is too big to be loaded into RAM, then GPU and fed to a CNN at once. Therefore, I implemented a data generator filling out the blueprint I found in a tutorial for data generator use in Keras. The data generator facilitates the loading of MRI scans 'on multiple cores in real time and feed[ing] it right away to your deep learning model' (Amidi 2017). Furthermore, it provided a frame work to contain all my preprocessing steps.

Using Amira ZIB Edition Version 2018.10 (Stalling et al. 2005) on a variety of over 50 MRI scans, I identified the range of slices from 25 to 135 from a total of 160 slices to be the highly visible range, i.e. visibly displaying the knee joint in all three orientation planes. This was even the case in MRI scans that showed the knee in a slightly shifted position probably due to inconsistencies in recording. In every orientation plane, the information of slices near one another is redundant since they show much of the same physiological features. A slice distance of 10 to 30 slices from a total of 160 slices was observed to ensure enough variety as well as to not miss important anatomical structures.

To reduce image preprocessing and thereby human bias to a reasonable amount, the original dimensions were kept whenever possible. Axial planes are counted from proximal to distal, coronal planes from anterior to posterior and sagittal planes from proper right to proper left.

3 2D One Slice Approach

The simplest method was fine-tuning a *S2DCNN* pretrained on planar non-medical images by feed-forwarding one MRI slice through the network and then applying backpropagation to the non-convolutional layers. I used the pretrained VGG19 network (Simonyan & Zisserman 2014). This CNN was developed for and came in second place at the 2012 ILSVRC (Russakovsky et al. 2015). It has a classical CNN architecture as described in the Introduction. I chose VGG19 due to its architectural simplicity and good performance in comparison to other 2D CNNs (Krizhevsky et al. 2012; Szegedy et al. 2015; He et al. 2016).

Since the CNN is fed with only one two-dimensional input image, it requires to choose manually which slice and plane (given sagittal plane or transforming it into coronal or axial plane) the CNN is initialized with. This choice filters the information input to the network and thereby already assumes that the selected slice is representative of the laterality of the whole MRI scan. To decide which plane and slice works best, I did a test series with reduced scope. The configuration with the best result was then trained properly.

3.1 Network Architecture of VGG19

The pretrained VGG19 network was retrieved from the Keras library (Simonyan & Zisserman 2014). The convolutional layers of the net used the weights pretrained on ImageNet (Deng et al. 2009; Russakovsky et al. 2015). The other layers' weights were initialized pseudo-randomly.

The architecture of VGG19 consists of 19 weight layers organized in five convolutional and one fully connected block (Fig. 3.1). The convolution's kernels all have size (3×3) and ReLU is used as activation function in all layers except for the output layer. The number of filters in a convolutional layer, as well as the number of convolutional layers in a block, increases throughout the network from input to output layer. At the end of every convolutional block, a maxpooling operation is performed by a kernel of (2×2) voxels. After the last convolutional block, the output tensor is flattened and fed forward through two ReLU-activated fully connected layers of size 4096 and one softmax-activated output layer of size two. For more details please see Fig. 3.1. Compared to other CNNs (Krizhevsky et al.

2012; Szegedy et al. 2015; He et al. 2016), VGG19 has many parameters and is moderately deep.

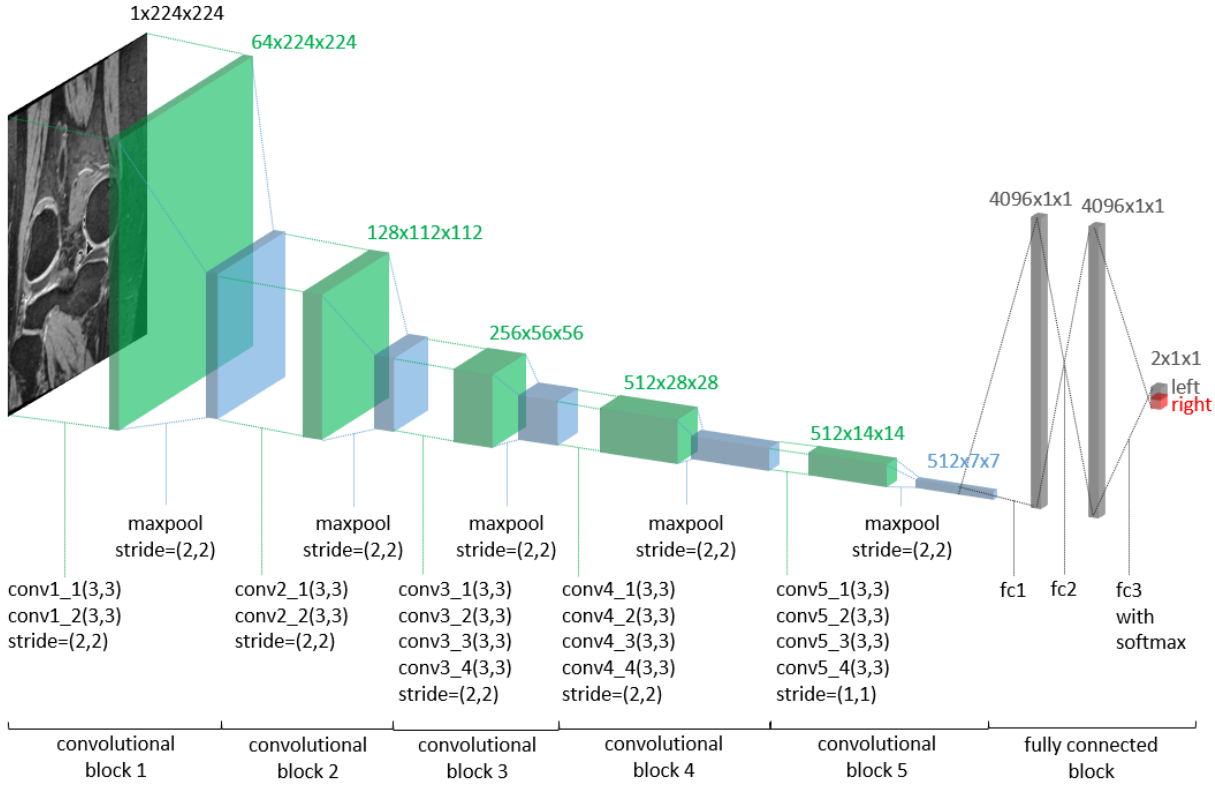


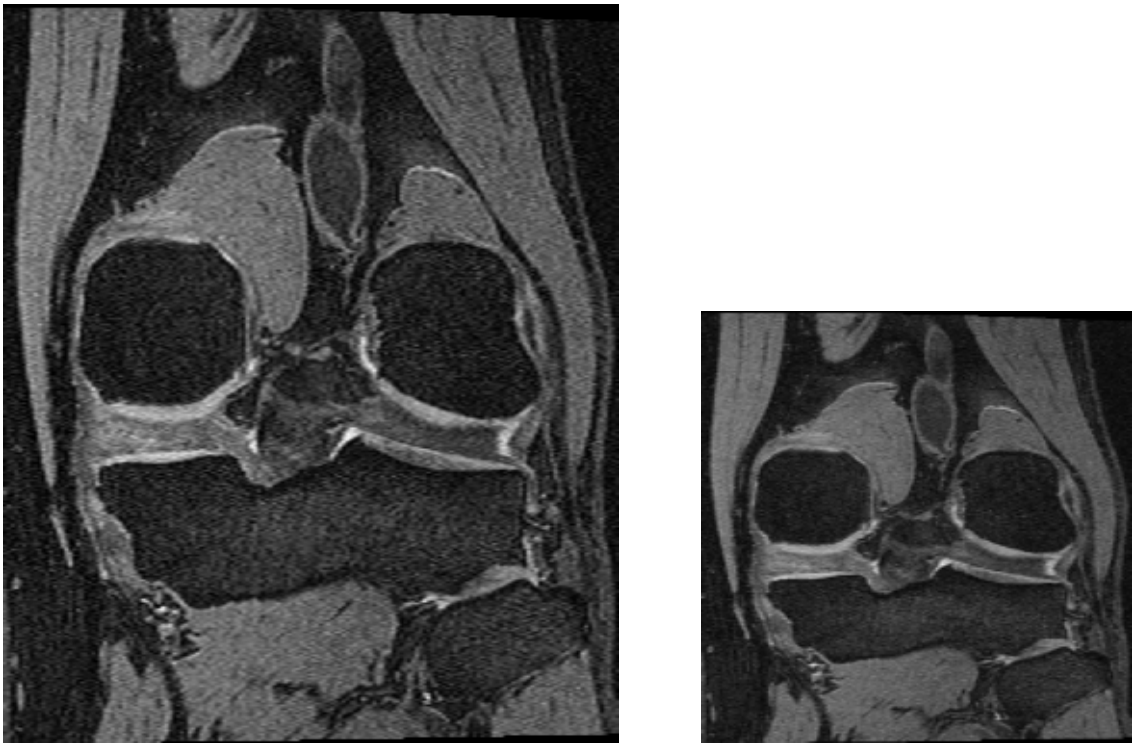
Figure 3.1: Architecture of VGG19.

3.2 Image Loading and Preprocessing for VGG19

As input, VGG19 takes a batch of two-dimensional RGB images with dimensions ($224 \times 224 \times 3$). To conform to this requirement, the raw MRI scan was transposed to fit the desired plane in preprocessing. Then it was squeezed and resized to dimensions ($160 \times 224 \times 224$) in two steps using the bilinear interpolation resize function from Open-cv: First, along the last two axes, second, along the first two axes. The resulting array was converted to RGB with an Open-cv function that triples the grayscale intensity value, and normalized using L2-norm with a Numpy function (Fig. 3.2). After this, the desired slice was extracted and saved as future input or directly fed to the network.

In the context of my test series, preprocessing was carried out during training time because every slice was only used once during training and, therefore, total temporal advantages for preprocessing before training time would have been minimal to non-existing.

In contrast, for training more thoroughly on the best configuration of the test series, all MRI scans were preprocessed before training and the desired slice was extracted and saved as a npy file. This preprocessing took about six hours on one CPU. During training the slice was loaded from npy file, improving training time by avoiding repeated preprocessing of the same MRI scan in every epoch.



(a) Before preprocessing (size 320x384) (b) After preprocessing (size 224x224)

Figure 3.2: Exemplary comparison of one slice (coronal-105) of the original (a) and the corresponding slice in the preprocessed image (b) for VGG19.

3.3 Grid Search for Preselection of Input

To be able to make an informed choice of input slice, I compared fifteen different input image configurations in a grid search, more precisely, five different slices from each of the three planes. I limited the search space because training one network on each of the $\geq 3 \cdot 160$ possible slices from all three orientation planes would have been too time-consuming. Moreover, an exhaustive search space is neither necessary nor sensible due to information redundancy in neighboring slices. I chose the five slices in each plane evenly distributed from the highly visible range $(25, 135)$, resulting in working with slices 30, 55, 80, 105 and 130 (see Fig. 3.3).

For each of the fifteen input configurations, the network was trained partially on 40 batches each consisting of 70 MRI image slices from the respective plane and position. This corresponds to about 41% of the training set. Training aimed at minimizing binary cross entropy loss using stochastic gradient descent with momentum 0.9 as in the original paper (Simonyan & Zisserman 2014). Starting with a learning rate of 0.01, like Simonyan et al., led to fast divergence, as did 0.005. Thus, I used an initial learning rate of 0.001. I reinitialized the weights with the same values every time before training on another configuration. During training the order of MRI scans was shuffled in every training epoch pseudo-randomly, but always setting the same seed ensured that the images were fed into every network in the same order leading to greater comparability. The training part of the grid search took on average 1:45 hours per configuration and about 26 hours in total. Then all trained CNNs were evaluated using the validation set, which took on average another 32 minutes per configuration and eight hours in total.

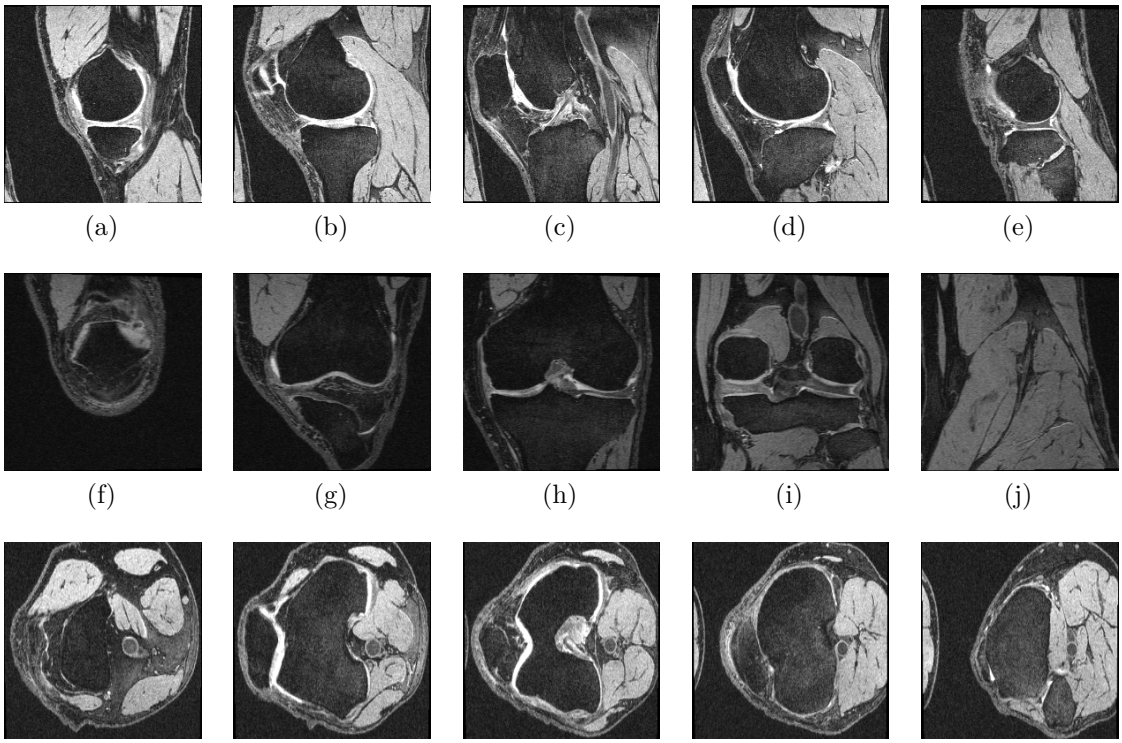


Figure 3.3: MRI slices used for the test series: sagittal slices (a) 30, (b) 55, (c) 80, (d) 105 and (e) 130; coronal slices (f) 30, (g) 55, (h) 80, (i) 105 and (j) 130; axial slices (k) 30, (l) 55, (m) 80, (n) 105 and (o) 130.

As can be seen in Table 3.1, the CNN converged on all but one input configuration. Four configurations even led to a validation accuracy of 100%. The only input that did not lead to convergence of CNN was the slice 80 of the sagittal plane. This should not be surprising as this slice represents the middle sagittal plane in each knee and is therefore anatomically the same in left and right knee joint. In the best performing input configurations, all images showed at least one laterality-specific anatomical feature: The coronal-80 and the coronal-105 slices both show the asymmetric condyles of the tibiofemoral joint. Additionally, the coronal-105 slice's view includes a small part of the fibular epiphysis. The axial-130 slice features cross sections of tibia and fibula. Similarly, the sagittal-30 slice shows the sagittal plane next to the tibia with the tibiofemoral joint in the right knee. It should be noted that the sagittal-130 slice in one knee is anatomically identical to the sagittal-30 in the other knee. Its slightly inferior performance may be explained by the limited and different training instances that it has been shown in the grid search. Although they were from the same original MRI, the instances each featured different slices and were therefore not identical.

Plane	Slice (see fig. 3.3)	Validation Accuracy [%]	Validation loss
sagittal	30 (a)	**100.00**	0.014191
	55 (b)	99.47	0.043422
	80 (c)	50.21	0.928622
	105 (d)	99.95	0.025857
	130 (e)	99.89	0.017925
coronal	30 (f)	96.14	0.166646
	55 (g)	99.89	0.029023
	80 (h)	**100.00**	0.016887
	105 (i)	**100.00**	0.014808
	130 (j)	99.95	0.026045
axial	30 (k)	99.89	0.042842
	55 (l)	99.89	0.031137
	80 (m)	99.31	0.064924
	105 (n)	99.20	0.058969
	130 (o)	**100.00**	0.019104

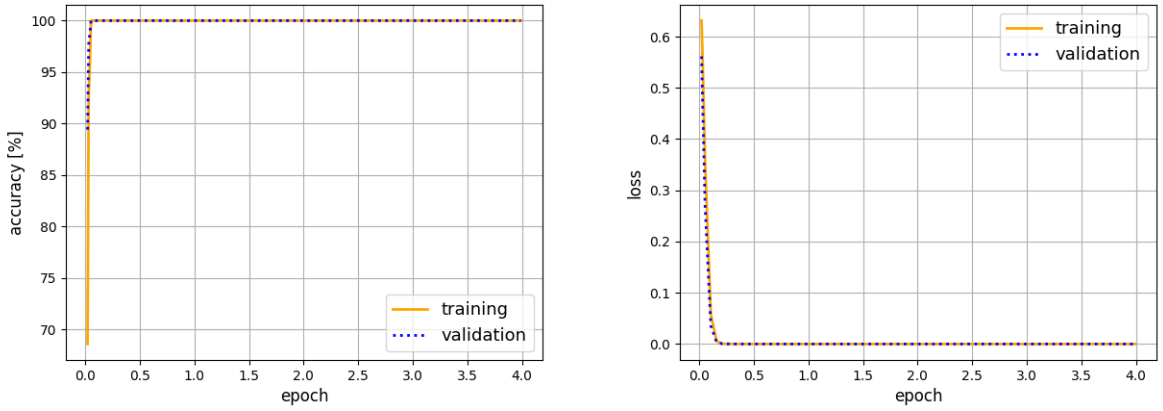
Table 3.1: Results from validation of the *S2DCNN* test series.

3.4 Training of VGG19

From the configurations examined in the grid search that had achieved 100% accuracy, I picked one as input to train the CNN again with the same optimizer, loss function, learning rate and batch size, this time on all data, and asses it in detail. For this, I chose the coronal-105 slice because it was the only one that included two laterality-specific anatomical structures. I trained VGG19 on four epochs of 84 batches with 70 images each, validating performance on the validation set after each of the first five and after every five batches. This took about 20 minutes. As it is wide-spread in analyzing simple 2D CNNs, the activations of the trained network in the last convolutional layer were visualized using Keras-vis to create heatmaps with the input coronal-105 slices.

3.5 Results of Training VGG19

As shown in Fig. 3.4, the network's validation accuracy converged towards 100% after about five batches. Validation loss and accuracy converged approximately as fast as training accuracy and loss, indicating no overfitting. To ensure better comparability with the other approaches, I did not evaluate the model checkpoint after only 5 batches, but the first checkpoint after having trained on all training data, i.e. after the first epoch. At this checkpoint, VGG19 had a test accuracy of 100%.



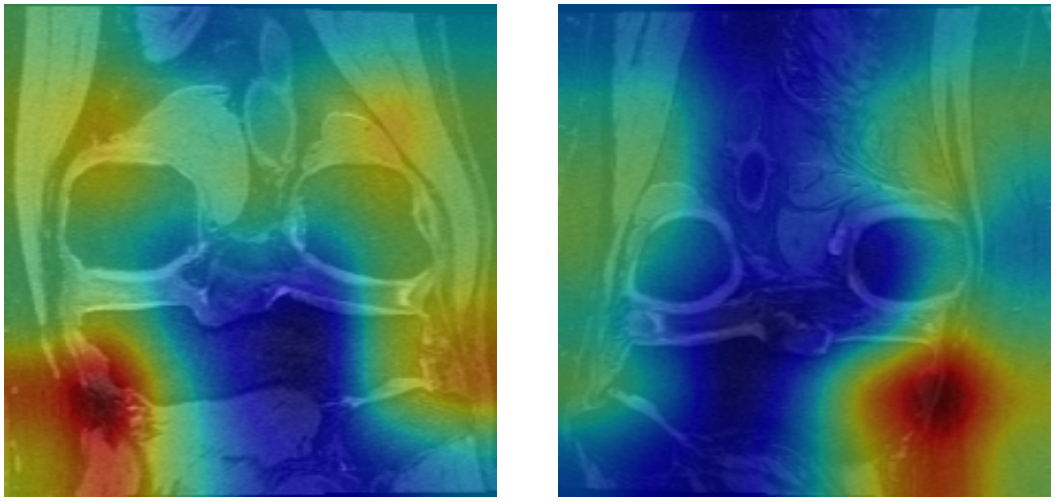
(a) Training progress of accuracy

(b) Training progress of loss

Figure 3.4: Convergence of loss and accuracy during training of VGG19.

The activation heatmap (Fig. 3.5) shows that VGG19 mainly uses the MRI scan's slice region where the absence of the fibula is expected to be, i.e. left bottom for a left knee

and right bottom for a right knee MRI scan, for classification. Additionally, the network activates the outer border areas of the femoral and tibial epiphyses. This illustrates that VGG19 indeed extracted the assumed laterality-specific anatomical features from the input slice.



(a) Left knee (b) Right knee

Figure 3.5: Activation heatmaps for trained VGG19 with exemplary coronal-105 slices.

4 2D Multiview Approach

The second two-dimensional method was a multi-view approach inspired by the *MVCNN* for 3D shape recognition published by (Su et al. 2015). Simply put, this approach is about 'learn[ing] to combine information from multiple views using a unified CNN architecture that includes a view-pooling layer' (Su et al. 2015). This network uses multiple simple 2D CNNs each being forward fed separately with one two-dimensional view of the three-dimensional image, producing an autonomous representation for each view. These single-view representations are aggregated at a view-pooling layer and then passed through another CNN combining all views for classification (see Fig. 4.1).

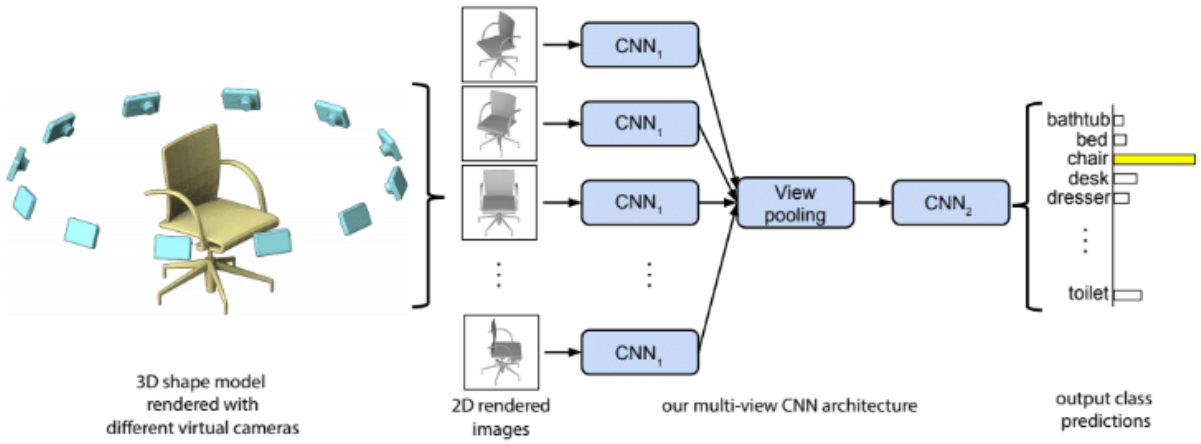


Figure 4.1: Different views of a 3D shape are passed through CNN 1 to extract view based features. These are then pooled across views and passed through CNN 2 to obtain a compact shape descriptor. (Su et al. 2015, Fig. 1).

In the context of my work, this technique regards the various MRI slices as different views. In the first step it analyzes all slices equally. More precisely, it extracts structure information only within the sliced planes of the input image ignoring the third dimension. This includes analyzing at least one slice with relevant information as manually selected in the *S2DCNN* approach (see Chapter 3). The second part now uses convolutions to gradually merge the information of the single slices. Since this is done using a CNN starting with a pooling layer, the way the slices in the original MRI scan were linked is respected in this step.

I worked with a TensorFlow- and Caffe-based python implementation (Lee 2018) by Tang Lee of the *MVCNN*, originally developed in MATLAB by Su et al. (2015) that I found cited in the original project's GitHub repository (Su 2019) (AlexNetMV). In contrast to Su's implementation that is based on VGG19, my tested model was built on AlexNet (Krizhevsky et al. 2012) as the default CNN that is arranged in parallel in the architecture to independently process the multiple views. AlexNet simplified handling the training of this model because it has less than half of the number of parameters of VGG19 and the resolution of MRI slices is slightly lower.

4.1 Network Architecture of AlexNetMV

Lee's AlexNetMV implementation differs slightly from the original implementation in terms of network architecture: One AlexNet CNN in the first part of the *MVCNN* consists only of convolutional and max pooling layers, passing on the max pooled convolutional layer's output tensor to the view pooling layer. In contrast, Su's implementation includes fully connected layers at the end of every CNN in the first part of the network and pools the predicted probabilities for each label of every view as a two-dimensional tensor in the pooling layer.

To go into detail of the network architecture (see Fig. 4.2), each one of the twelve parallelized AlexNets consists of three convolution blocks each containing convolutional layers activated by ReLU followed by a max pooling layer. The input layer is fed with an image of dimensions $(227 \times 227 \times 3)$. The first two blocks each have one, and the third block has three convolutional layers. The max pooling layers use a (3×3) kernel size and a stride of (2×2) voxels. The convolutional layers' number of kernels, as well as their stride, decreases from input towards view pooling layer while the size of the kernel increases. The max pooled output from each of the twelve CNNs is aggregated and again max pooled at a view pooling layer. Then the resulting tensor is passed through two fully connected inner layers with tensor size 4096 and the fully connected output layer with tensor size 2. All fully connected layers have a ReLU as activation function. The AlexNetMV's initial weights were already pretrained on ImageNet (Deng et al. 2009; Russakovsky et al. 2015).

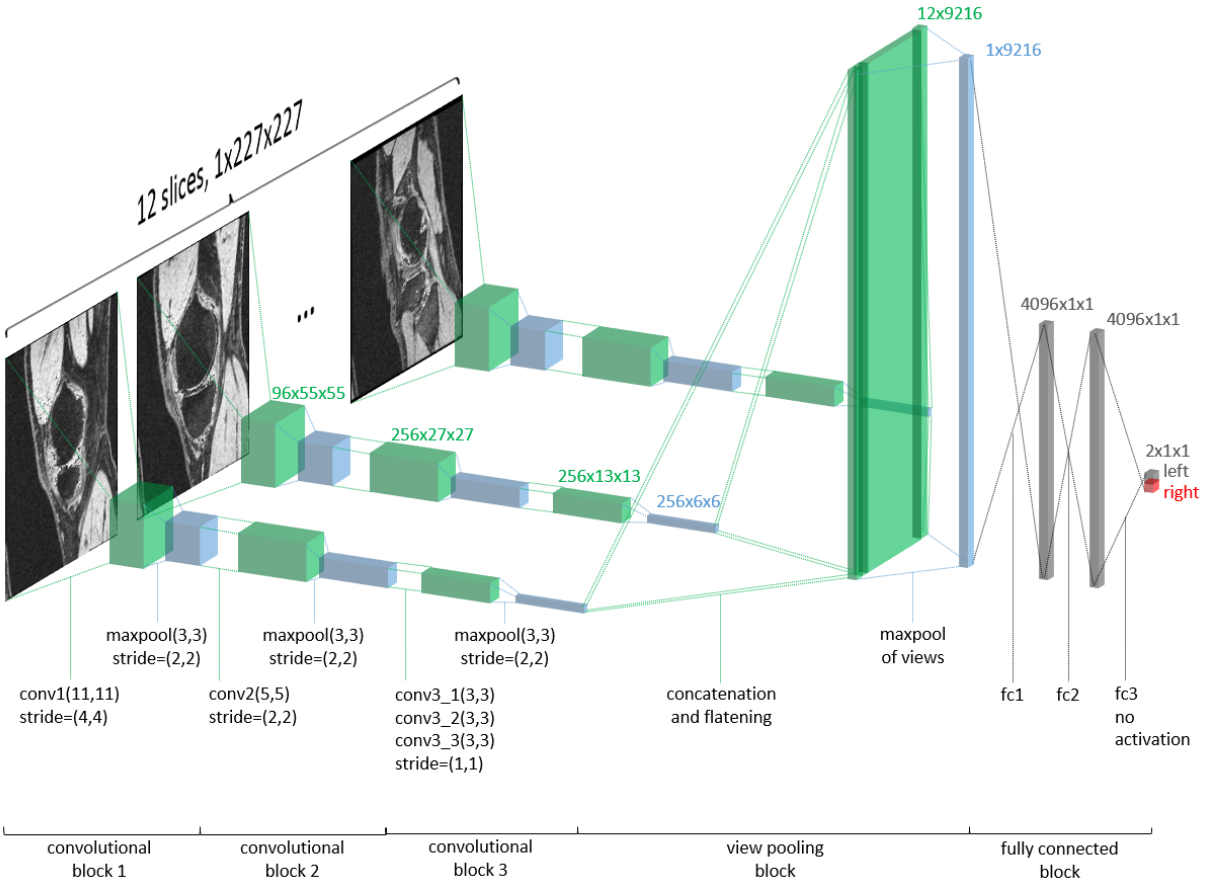


Figure 4.2: Architecture of AlexNetMV.

4.2 Image Loading and Preprocessing for AlexNetMV

AlexNetMV takes twelve RGB images, each of dimensions $(227 \times 227 \times 3)$, as input loaded from a png file. I used the data generator from my *S2DCNN* implementation to preprocess the raw MRI scan and comply to the input format in a preprocessing step before training. At first, it was squeezed and resized to dimensions $(160 \times 227 \times 227)$ with sagittal plane as first dimension in two steps along two planes and converted to RGB in the same manner as in the *S2DCNN* approach (see Fig. 4.3).

(a) Before preprocessing (size 384x384)

(b) After preprocessing (size 227x227)

Figure 4.3: Exemplary comparison of one slice of the original (a) and the corresponding slice in the preprocessed image (b) for AlexNetMV.

Then the slices 25, 35, 45, 55, 65, 75, 85, 95, 105, 115, 125 and 135 were saved in local storage creating a new smaller data base as input to AlexNetMV (see Fig. 4.4). I chose exactly twelve slices for input as in the original paper (Su et al. 2015) such that they were evenly distributed in the highly visible range $(25, 135)$. Note that this selection of slices does not contain the 80th sagittal slice that is anatomically identical in both knees and proved to be unsuitable for distinguishing between left and right in the VGG19 grid search (see Table 3.1). Preprocessing on one CPU took about 16:30 hours.

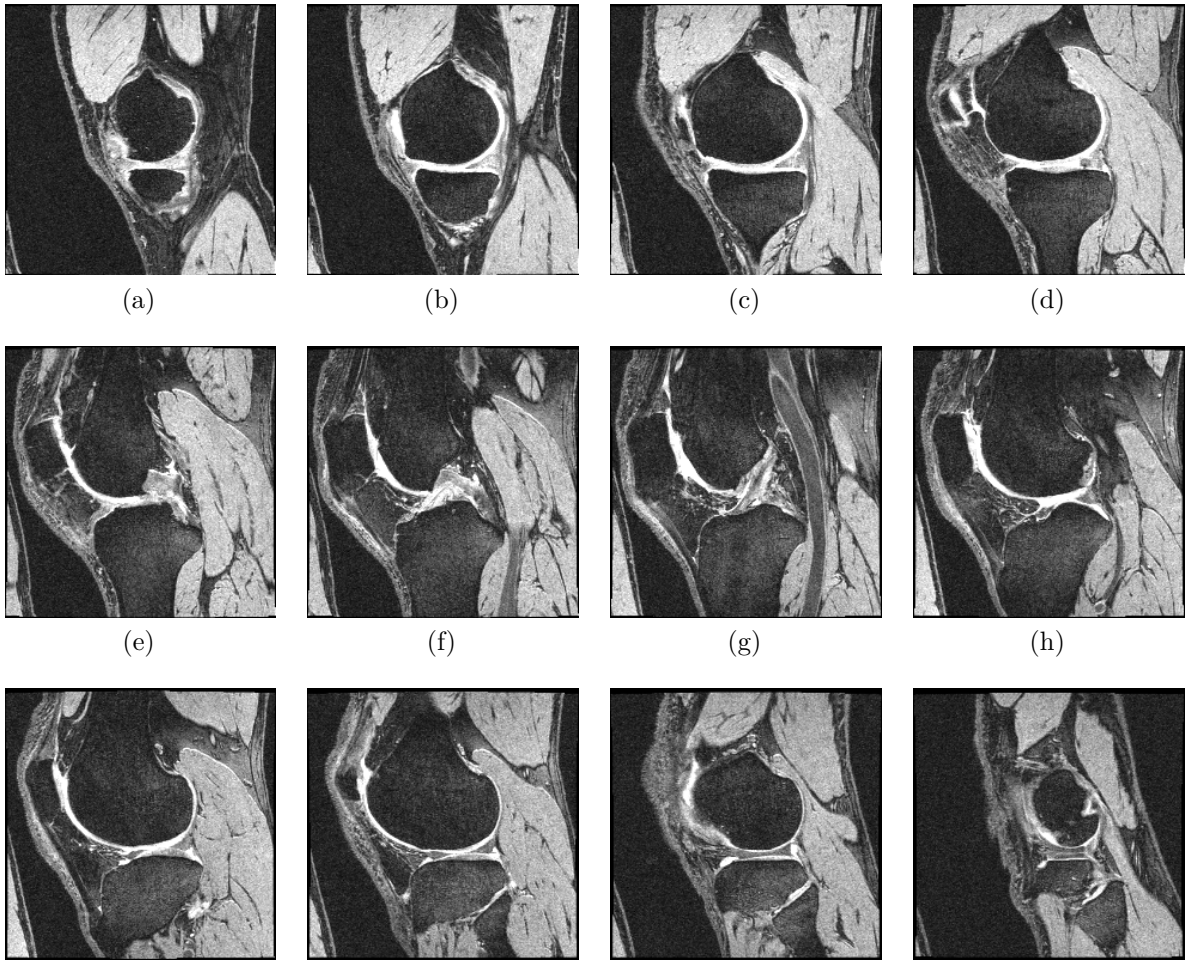


Figure 4.4: MRI slices used for AlexNetMV: sagittal slices (a) 25, (b) 35, (c) 45, (d) 55, (e) 65, (f) 75, (g) 85, (h) 95, (i) 105, (j) 115, (k) 125 and (l) 135.

4.3 Training of AlexNetMV

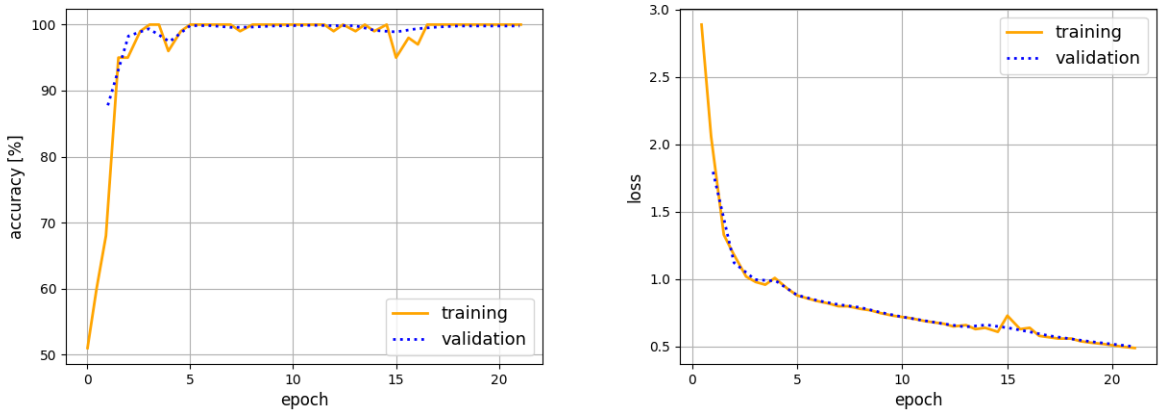
At first, I trained on a subset of 140 (training) + 40 (validation) MRI scans to see if the network started to converge. But there was no change in accuracy when training with different learning rates (0.1, 0.01, 0.001, 0.0001), different optimizers (Adam, Momentum and gradient descent optimizers) and different loss functions (cross entropy loss with sparse softmax and sigmoid function). Therefore, I removed the ReLU in the output layer as suggested in a similar issue post on GitHub (rlczddl & Potuaud 2017). This led to improvement of accuracy in the course of training on the subset. Experimenting with various validation points during training on the whole training set revealed a problem inherent to

the AlexNetMV code that leads to killing the process during a validation phase due to RAM overuse. This did not allow me training continuously for more than nine to eleven epochs nor validating at a different interval than on an epoch's end. Unfortunately, I could not reliably replicate this error. But when training was interrupted, I just resumed starting from the last model checkpoint.

I trained the AlexNetMV with Adam Optimizer and learning rate 0.0001 on 21 epochs with 66 batches of size 100, minimizing cross entropy loss with sparse softmax function. Training was stopped when the test and validation accuracy both started to decrease. I chose learning rate, optimizer and loss function as the defaults from the AlexNetMV project. Finally, training including validation took about 6:25 hours.

4.4 Results of Training AlexNetMV

As can be seen in Fig. 4.5, the network converged towards approximately 99.89% validation accuracy after eleven epochs. Afterwards, the loss continued to decrease, but validation accuracy started declining slightly and changing inconsistently. A decrease in learning rate after epoch eleven did not change this. When evaluating AlexNetMV from this step on the test set, it showed an accuracy of 99.79%. Analyzing the MRI scans from all three sets that could not be identified correctly, they did not appear to differ substantially from correctly classified MRI scans and did not share specific common qualities.



(a) Training progress of accuracy

(b) Training progress of loss

Figure 4.5: Convergence of loss and accuracy during training of AlexNetMV.

5 3D Convolution Approach

My third approach was the usage of a pretrained CNN with spatial convolutions (Maturana & Scherer 2015; Çiçek et al. 2016; Dubost et al. 2017; Wu et al. 2015; Zhu et al. 2017). This *3DCNN* takes a three-dimensional voxel grid as input and uses volumetric sliding windows to create multiple three-dimensional heat maps in their convolutional layers. This is followed by a pooling layer that downsamples the last convolutional layer's outputs 'along the spatial dimensions by replacing each [m^3] non-overlapping block of voxels with their maximum' (Maturana & Scherer 2015). Then multiple fully-connected layers combine the information extracted in the preceding layers.

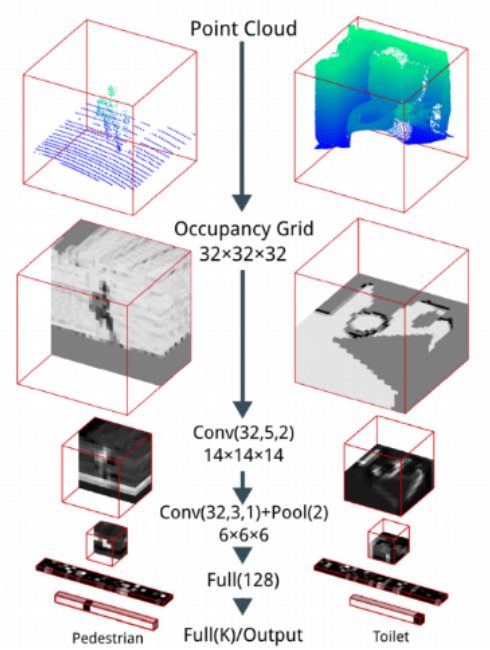


Figure 5.1: The VoxNet Architecture. 'Conv(f,d,s) indicates f filters of size d and at stride s, Pool(m) indicates pooling with area m, and Full(n) indicates fully connected layer with n outputs' (Maturana & Scherer 2015, Fig. 1).

In contrast to the two-dimensional approaches, the resolution and, consequently, the complexity of the input to the *3DCNN* is evenly distributed over three dimensions. Hence, for memory and computational reasons, the resolution of each two-dimensional slice, regardless

of its orientation plane, must be significantly smaller than in the two other approaches. To achieve this, I had to downsample the MRI scans considerably. This CNN then uses the downsampled volumetric data as input. As *3DCNN* implementation I selected VoxNet (Maturana & Scherer 2015) because of its simple architecture, superior performance to other *3DCNNs* and its relatively small number of weights. Originally, VoxNet was created for real-time object recognition e.g. on data RGBD cameras or range sensors.

5.1 Network Architecture of VoxNet

I used the TensorFlow implementation by Benjamin Kang Yue Sheng published on his GitHub (Sheng 2018). Sheng's VoxNet implementation was pretrained on the ModelNet40 data set (Princeton University 2019).

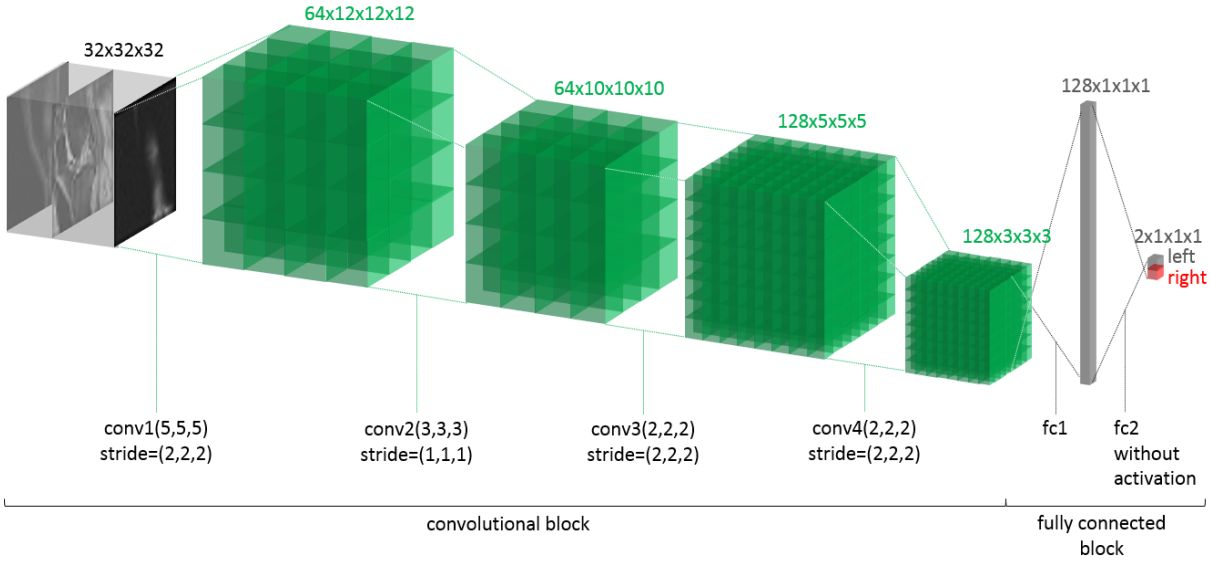


Figure 5.2: Architecture of VoxNet.

VoxNet's architecture consists of four 3D convolutional and two fully connected layers (Fig. 5.2). All layers except for the output layer employ batch normalization and ReLUs as activation functions. In the convolutional part of the network, the number of kernels increases from the input towards the fully connected layers, while the kernel size decreases. The last convolutional layer's output tensor is directly fed to the first fully connected layer. With fewer than one million weights, the network has a remarkably small number of parameters.

5.2 Image Loading and Preprocessing for VoxNet

VoxNet takes an occupancy grid with dimensions $(32 \times 32 \times 32)$ as input. The original VoxNet by Maturana et al. calculates its binary occupancy voxel grid input from point clouds (see Fig. 5.1). However, the concept of a binary occupancy grid does not make much sense for an MRI scan where most voxels are occupied, and the main information is in the different intensities representing different types of tissue, which classify different anatomical structures.

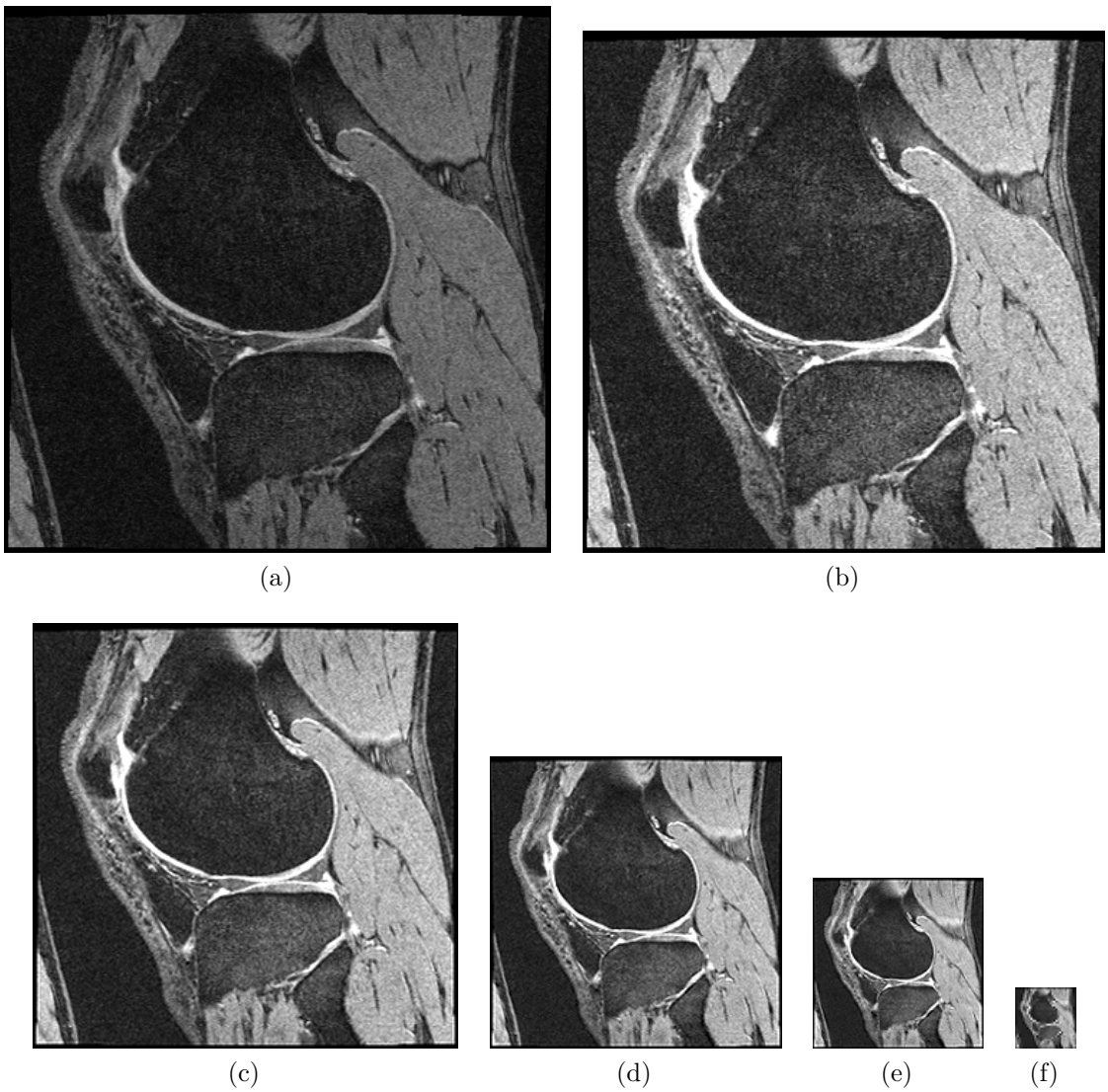


Figure 5.3: Slices from resizing steps on MRI scan: (a) shows slice sagittal-115 from raw MRI scan with dimensions $(160 \times 384 \times 384)$, followed by corresponding slices of resizing steps (b) 1 $(154 \times 364 \times 364)$, (c) 5 $(128 \times 288 \times 288)$, (d) 10 $(96 \times 192 \times 192)$, (e) 15 $(64 \times 96 \times 96)$ and finally (f) 20 $(32 \times 32 \times 32)$.

Therefore, I downsampled the raw MRI scans directly in a preprocessing step before training using the data generator from my simple 2D approach. For this approach, the resizing method employed in the preprocessings of the two-dimensional CNNs could not be used because the rapid downsampling to a much lower resolution let to a loss of all information resulting in a nearly black image. Instead, I squeezed and downsampled the MRI scan for VoxNet in 20 steps to an increasingly lower resolution. Each step was similar to the whole resizing operation in the two-dimensional CNN approaches: In each step I used the resize function from Opencv with bicubic interpolation over a (4×4) pixel neighborhood twice: First, along the last two axes, second, along the first two axes. The images in decreasing resolution can be seen in Fig. 5.3. The resized MRI scan (see Fig. 5.4) was then saved to disc for future use. Preprocessing all input data took about 25 hours.

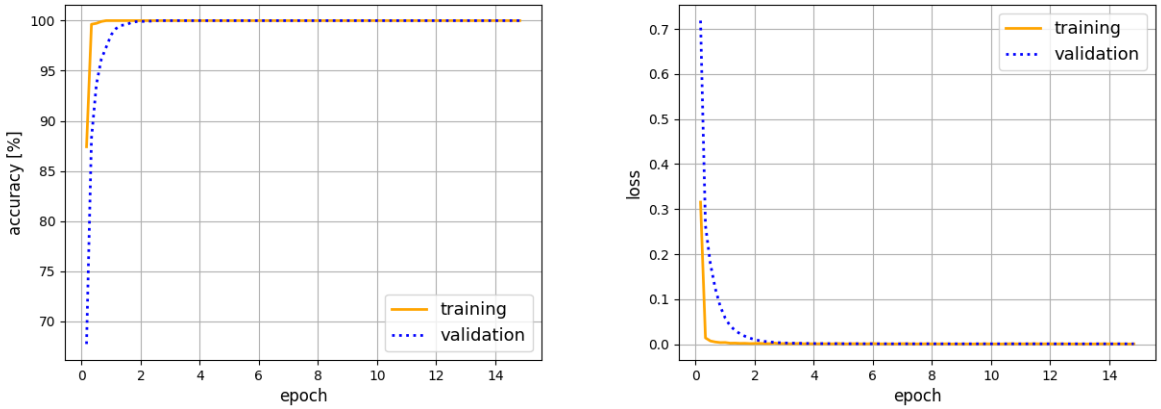
Figure 5.4: MRI sagittal slices in reduced resolution (32×32) used for VoxNet.

5.3 Training of VoxNet

VoxNet was trained on 15 epochs with 66 batches consisting of 100 images each, using Adam optimization to minimize softmax cross entropy loss. The initial learning rate was 0.001. Stochastic gradient descent optimized L2 loss for weight decay. The current network's performance on the validation data set was evaluated every eleven batches. The implementation decayed the learning rate gradually at every validation step. Training was stopped when the decrease in validation loss stagnated. The entire training process took about six minutes, with 2.5 seconds per epoch on average.

5.4 Results of Training VoxNet

As can be seen in Fig. 5.5, VoxNet's validation accuracy converged to 100% after three epochs. Training accuracy converged faster than validation accuracy and reached 100% during the first epoch after having been presented with only 66.67% of the training data set. Similarly, training loss converged faster than training loss.



(a) Training progress of accuracy

(b) Training progress of loss

Figure 5.5: Convergence of loss and accuracy during training of VoxNet.

Learning rate decay was steep and approximately linear until after the eighth epoch (Fig. 5.6). When evaluating the network's model checkpoint after the fifth training epoch on the test set, it had an accuracy of 100%.

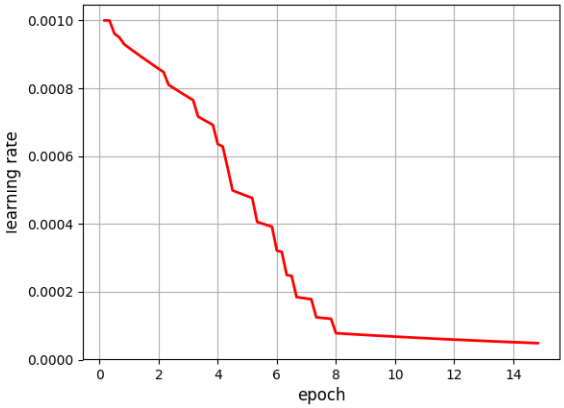


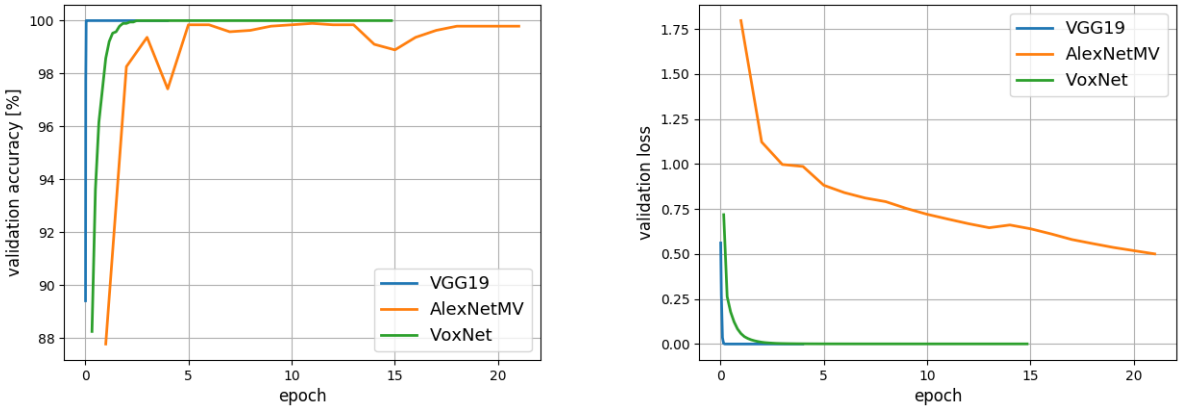
Figure 5.6: Decay of learning rate during training of VoxNet.

6 Discussion and Conclusion

The motivation behind the comparison of different CNNs for classifying knee MRI scans in terms of laterality was to find a method that could be included in the knee joint segmentation pipeline, to hopefully make the algorithm more efficient, faster, and less data dependent. The most important requirement for a classification method in this context is having a high accuracy because subsequent steps in the pipeline depend on its predictions. Furthermore, it is interesting to compare the total time of preprocessing and classifying an input MRI scan with a trained model since this may affect evaluation and training durations of the segmentation process.

6.1 Comparison of VGG19, AlexNetMV and VoxNet

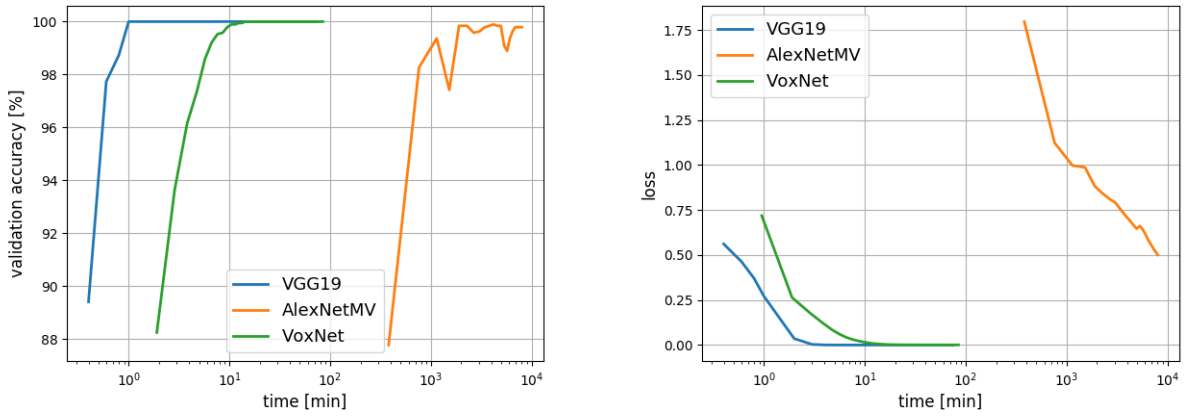
VGG19 and VoxNet both achieved test, training and validation accuracies of 100%, meaning they could classify every MRI scan in the data set correctly. AlexNetMV also achieved a training accuracy of 100%, but it was less stable over different epochs. More importantly, its validation and test accuracies only reached 99.79% in converged state.



(a) Validation accuracy

(b) Validation loss

Figure 6.1: Comparison of training progress over number of epochs.



(a) Validation accuracy (b) Validation loss

Figure 6.2: Comparison of training progress over time.

As can be seen in Fig. 6.1, relative to the number of epochs, VGG19 was the fastest to converge, followed by VoxNet. Figure 6.2 shows that relative to time, VoxNet converged nearly as fast as VGG19. From both perspectives, AlexNetMV was the slowest to converge. This may be due to its lower initial learning rate compared to the other two approaches (see Fig. 6.3).

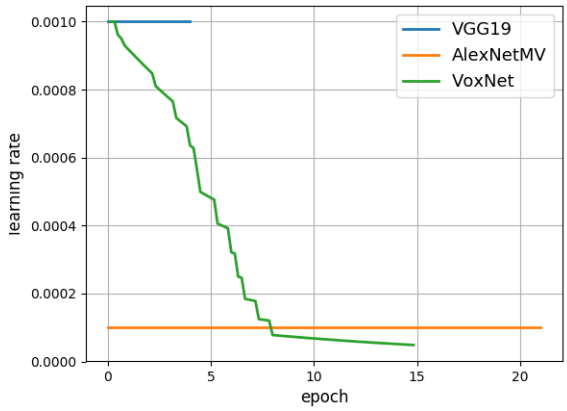


Figure 6.3: Comparison of learning rate over number of epochs.

As can be seen in Table 6.1, duration of preprocessing is similar for VGG19 and AlexNetMV, while it takes more than twice as long for VoxNet. Evaluation is fastest in VGG19 and approximately twice as slow in AlexNetMV and VoxNet.

The chosen implementations differ substantially in terms of depth and number of parame-

ters: VGG19 is the deepest model with the most parameters. AlexNetMV is approximately half as deep and big in parameter size. VoxNet is only 25% less deep than AlexNetMV but has considerably less parameters.

	VGG19	AlexNetMV	VoxNet
Test accuracy [%]	100.00	99.79	100.00
Validation accuracy [%]	100.00	99.79	100.00
Training Accuracy [%]	100.00	100.00	100.00
Time preprocessing/scan [s]	2.4060	2.4946	6.1152
Time evaluating/scan [s]	0.0613	0.0518	0.0532
Time preprocessing + evaluating/scan [s]	2.4270	2.5464	6.1684
Epochs till convergence	<1	11	3
Time till convergence	18s	>3h	7.5s
Initial learning rate	0.001	0.0001	0.001
Input resolution	$(224 \cdot 224)$ = 50,176	$(12 \cdot 227 \cdot 227)$ = 618,348	$(32 \cdot 32 \cdot 32)$ = 32,768
Relative resolution after preprocessing [%]	0.213	2.621	0.139
Network depth (#layers with weights)	19	8	6
Number parameters [million]	143	85	<1

Table 6.1: Comparative values.

The preprocessing for all three approaches reduces the resolution of the MRI scan drastically. The input resolution of AlexNetMV is the highest, VGG19's and VoxNet's are smaller by a factor of about 10 and 20, respectively. In the preprocessing for VoxNet, the complexity of input data is reduced approximately equally along all three axes, whereas in VGG19 and AlexNetMV, complexity is drastically reduced along the first axis and only cut roughly in half along the other two axes. Another notable difference is that complexity reduction in VoxNet is completely automatic, while partly and completely manually decided in AlexNetMV and VGG19, respectively.

6.2 General Assessment

For classifying knee MRI scans in terms of laterality, both VGG19 and VoxNet perform very well and are clearly superior to AlexNetMV. From these two best performing networks, VGG19 has the advantage of converging after having been trained on only a small part of

the training set which would still make it qualified if less data were available. In contrast, VoxNet needs more data and more iterations but nonetheless approximately the same time for the same result. However, preprocessing for VoxNet is more time-consuming, resulting in longer total evaluation times. But since these characteristics are only secondary for the task assigned to the CNNs here, VGG19 and VoxNet can be both considered to be best implementations in the context of this thesis.

This is mostly also true for utilization within the Therapy Planning Research Group's segmentation pipeline due to their perfect accuracy, which allows subsequent steps in the pipeline to depend on the classification result. However, besides accuracy, results on total evaluation time including preprocessing that favor VGG19 should also be taken into account since they influence general training and evaluation time. Consequently, VGG19 is slightly preferable for use in the segmentation pipeline.

Laterality is a type of information that can be found throughout the whole input image because the human body is, with the exception of inner organs, symmetric to the median plane and therefore every left or right body part is in itself asymmetrical. I strongly suspect that this extensive information distribution in the input is the reason for the *S2DCNN* implementation's impressive performance independent from the chosen view. This characteristic quality of laterality is probably also the cause for all non-medial slices performing comparably well in the grid search for VGG19. Based on this observation, I argue that other classifications of features that share this quality in the input data can be solved equally well with approaches from the *S2DCNN* category.

In my employment of the *S2DCNN* implementation, a manual complexity reduction in form of slice selection was necessary. Such a manual selection may reduce the extent to which the method can be used within a different context, e.g. as a part of a pipeline that is not the first element or reusing the model in the context of another MRI routine or a different body part. However, if a classification task uses information distributed extensively in the input image, this selection should not make a big difference in results.

3DCNNs are also an adequate choice for solving a general classification task on medical 3D data, as the extraordinary performance of VoxNet proves. In contrast to *S2DCNNs*, these networks have the advantage of not needing manual complexity reduction and therefore their input contains the whole information which makes them preferable as one building block in the middle of a pipeline or if an even distribution of information in the whole 3D image is uncertain. With the increasing capacity of GPUs, downsampling to low resolution will be probably less and less necessary in the future.

I think it is noteworthy that in this comparison the worst performing model also had the

most complicated architecture and the highest input data resolution. This may be due to feeding the network not only with more information at higher resolution, but potentially also with more noise or redundant information, increasing entropy in the input. Similarly, Qi et al. (2016) attributed the observed superiority of *MVCNN* to a more straight-forward model approach. Moreover, these result is in accordance with Occam's razor since they favor the simpler models.

7 Outlook

I found two methods with phenomenal accuracy in classification of knee MRI laterality that can be used as a tool before or within the segmentation process to make it more efficient, faster and less data dependent, since MRI scans with laterality labels are available in a substantially larger number than manually segmented ones. The more reliable this segmentation, the better the biomarkers can be extracted and used for further research and diagnosis of OA. It is reasonable to assume that similar tools for other additional features, such as omics data obtained from patients' biospecimen in the OAI study, can be developed and included in segmentation and diagnosis pipelines to improve them.

Further work may study different implementation representations of the three main categories to verify the results and refine the conclusions. Additionally, alternative input projections, e.g. rotations around the three middle axes from each orientation plane or rotated perspectives on a segmented knee bone structure around the 3D image, could be tested as input to *S2DCNN* and *MVCNN*. This may result in input data that represents the laterality even better, because laterality-specific anatomical structures that are not within one anatomical plane could still be within one input image. Other ML approaches, such as support vector machines or massive training networks (Suzuki 2017), could also be studied and compared to investigate this thesis' bias of preference for CNNs. Since all investigated CNNs reached a very high accuracy score, it would be interesting to examine the dependency of the approaches' performance on the amount of training data. I used knee MRI scans from the OAI that were within a certain norm. To continue, one could examine the accuracy of the methods with input data outside this norm, e.g. from children or younger adults, with fractures, other injuries or medical implants to analyze external validity.

This thesis generally studied approaches on three-dimensional MIA with CNNs for a very specific task. The very well-performing CNNs, namely the *S2DCNN* implementation VGG19 and the *3DCNN* implementation VoxNet, have the potential to be used as a part of bigger classification or segmentation pipelines for medical images since it can be assumed that they can solve the laterality classification task similarly on other body parts as described above. Furthermore, the *3DCNN* approach can be easily adjusted to more complex tasks

by increasing input resolution and architectural complexity, e.g depth. Then a *3DCNN* could also extract information that is more locally concentrated, e.g. fractures or other irregularities, and therefore cannot be represented properly in a low-resolution voxel grid. Such computer-aided methods in medicine have become more and more common and will in all probability continue to become even more widespread for diagnosis, therapy planning and treatment in the future (Choy et al. 2018).

List of Figures

List of Tables

Bibliography

Amazon. (2019). *Amazon Machine Learning Developer Guide.* Retrieved 2019-01-05, from https://docs.aws.amazon.com/machine-learning/latest/dg/splitting-the-data-into-training-and-evaluation-data.html

Ambellan, F., Tack, A., Ehlke, M., & Zachow, S. (2019). Automated Segmentation of Knee Bone and Cartilage combining Statistical Shape Knowledge and Convolutional Neural Networks: Data from the Osteoarthritis Initiative. *Medical Image Analysis*, *52*(2), 109-118. doi: 10.1016/j.media.2018.11.009

Amidi, S. (2017). *Keras Data Generator Tutorial from Stanford University.* Retrieved 2019-01-05, from https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly

Basaia, S., Agosta, F., Wagner, L., Canu, E., Magnani, G., Santangelo, R., & Filippi, M. (2018). Automated Classification of Alzheimer's Disease and Mild Cognitive Impairment using a Single MRI and Deep Neural Networks. *NeuroImage: Clinical*. Retrieved from http://www.sciencedirect.com/science/article/pii/S2213158218303930 doi: 10.1016/j.nicl.2018.101645

Bover-Ramos, F., Viña Almunia, J., Cervera-Ballester, J., Penarrocha, M., & Mira, B. (2018). Accuracy of Implant Placement with Computer-Guided Surgery: A Systematic Review and Meta-Analysis Comparing Cadaver, Clinical, and In Vitro Studies. *The International Journal of Oral & Maxillofacial Implants*, *33*, 101-115. doi: 10.11607/jomi.5556

Chen, L., Wu, Y., DSouza, A. M., Abidin, A. Z., Wismüller, A., & Xu, C. (2018). MRI Tumor Segmentation with Densely Connected 3D CNN. In *Medical Imaging: Image Processing* (Vol. 10574, p. 10574-10574). Retrieved from https://doi.org/10.1117/12.2293394 doi: 10.1117/12.2293394

Choy, G., Khalilzadeh, O., Michalski, M., Do, S., Samir, A. E., Pianykh, O. S., ... Dreyer, K. J. (2018). Current Applications and Future Impact of Machine Learning in Radiology. *Radiology*, *288*(2), 318-328. Retrieved from https://doi.org/10.1148/radiol.2018171820 doi: 10.1148/radiol.2018171820

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., & Ronneberger, O. (2016). 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In *Medical Image Computing and Computer-Assisted Intervention* (p. 424-432). Springer International Publishing. doi: 10.1007/978-3-319-24574-4_28

Cornell University. (2019). *Arxive E-Print Service.* Retrieved 2019-01-05, from `https://arxiv.org/`

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (p. 248-255). doi: 10.1109/CVPR.2009.5206848

Doi, K. (2007). Computer-Aided Diagnosis in Medical Imaging: Historical Review, Current Status and Future Potential. *Computireized Medical Imaging and Graphics*, *31*, 198-211. doi: 10.1016/j.compmedimag.2007.02.002

Dubost, F., Bortsova, G., Adams, H. H. H., Ikram, M. A., Niessen, W. J., Vernooij, M. W., & de Bruijne, M. (2017). GP-Unet: Lesion Detection from Weak Labels with a 3D Regression Network. In *Medical Image Computing and Computer-Assisted Intervention* (p. 214-221). Springer International Publishing. doi: 10.1007/978-3-319-66179-7_25

Eckstein, F., Kwoh, C. K., & Link, T. M. (2014). Imaging Research Results from the Osteoarthritis Initiative (OAI): a Review and Lessons Learned 10 Years After Start of Enrolment. *Annals of the Rheumatic Diseases*, *73*(7), 1289-1300. doi: 10.1155/2011/475684

Eckstein, F., Wirth, W., & Nevitt, M. C. (2007). Recent Advances in Osteoarthritis Imaging-The Osteoarthritis Initiative. *Nature Reviews Rheumatology*, *8*, 622-630. Retrieved from `https://doi.org/10.1038/nrrheum.2012.113` doi: 10.1038/nrrheum.2012.113

Engle, R. L. (1992). Attempts to Use Computers as Diagnostic Aids in Medical Decision Making: A Thirty-Year Experience. *Perspectives in Biology and Medicine*, *35*(2), 207-219. Retrieved from `http://muse.jhu.edu/journals/perspectives_in_biology_and_medicine/v035/35.2.engle.html` doi: 10.1353/pbm.1992.0011

European Musculoskeletal Conditions Surveillance and Information Network. (2012). *Musculoskeletal Health in Europe: Report v5.0.* Retrieved 2019-01-28, from `http://www.eumusc.net/myUploadData/files/Musculoskeletal%20Health%20in%20Europe%20Report%20v5.pdf`

Freeman, W., Perona, P., & Schölkopf, B. (2008). Guest Editorial. *International Journal of Computer Vision*, *77*(1). doi: 10.1007/s11263-008-0127-7

Fuchs, J., Kuhnert, R., & Scheidt-Nave, C. (2017). 12-Monats-Prävalenz von Arthrose in Deutschland. *Journal of Health Monitoring*, *2*(3), 15-25. doi: 10.17886/RKI-GBE-2017-054

Geron, A. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (1st ed.). O'Reilly Media, Inc.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. (`http://www.deeplearningbook.org`)

Google Deepmind. (2019). *Deepmind AlphaGo.* Retrieved 2019-01-05, from `https://deepmind.com/research/alphago/`

Google Machine Learning. (2019). *Machine Learning Crash Course.* Retrieved 2019-01-05, from `https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data`

Google Scholar. (2019). *Google Scholar Search Engine.* Retrieved 2019-01-05, from `https://scholar.google.de/`

Gupta, S., Girshick, R., Arbeláez, P., & Malik, J. (2014). Learning Rich Features from RGB-D Images for Object Detection and Segmentation. In *Computer Vision – ECCV 2014* (p. 345-360). Springer International Publishing. doi: 10.1007/978-3-319-10584-0_23

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 770-778. doi: 10.1109/CVPR.2016.90

Helmick, C. G., Felson, D. T., Lawrence, R. C., Gabriel, S., Hirsch, R., Kwoh, C. K., ... Stone, J. H. a. (2008). Estimates of the Prevalence of Arthritis and Other Rheumatic Conditions in the United States: Part I. *Arthritis & Rheumatism*, *58*(1), 15-25. Retrieved from `https://onlinelibrary.wiley.com/doi/abs/10.1002/art.23177` doi: 10.1002/art.23177

Hernandez, D., Garimella, R., Eltorai, A. E. M., & Daniels, A. H. (2017). Computer-assisted Orthopaedic Surgery. *Orthopaedic Surgery*, *9*(2), 152-158. Retrieved from `https://onlinelibrary.wiley.com/doi/abs/10.1111/os.12323` doi: 10.1111/os.12323

Jacobs, C., van Rikxoort, E., Scholten, E., A de Jong, P., Prokop, M., Schaefer-Prokop, C., & van Ginneken, B. (2014). Solid, Part-Solid, or Non-Solid? Classification of Pulmonary Nodules in Low-Dose Chest Computed Tomography by a Computer-Aided Diagnosis System. *Investigative Radiology*, *50*, 168-173. doi: 10.1097/RLI.0000000000000121

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-Scale Video Classification with Convolutional Neural Networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition* (p. 1725-1732). doi: 10.1109/CVPR.2014.223

Kasai, S., Li, F., Shiraishi, J., Li, Q., & Doi, K. (2006). Computerized Detection of Vertebral Compression Fractures on Lateral Chest Radiographs: Preliminary Results with a Tool for Early Detection of Osteoporosis. *Medical Physics*, *33*(12), 4664-4674. Retrieved from `https://aapm.onlinelibrary.wiley.com/doi/abs/10.1118/1.2364053` doi: 10.1118/1.2364053

Khvostikov, A., Aderghal, K., Benois-Pineau, J., Krylov, A. S., & Catheline, G. (2018). 3D CNN-based Classification using sMRI and MD-DTI Images for Alzheimer Disease Studies. *Computing Research Repository*, *abs/1801.05968*. Retrieved from `http://arxiv.org/abs/1801.05968`

Krizhevsky, A., Sutskever, I., & E. Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*, *25*, 1106-1114. doi: 10.1145/3065386

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., & Jackel, L. D. (1990). Handwritten Digit Recognition with a Back-Propagation Network. In *Advances in Neural Information Processing Systems 2* (p. 396-404). Morgan-Kaufmann. Retrieved from `http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf`

Lee, T. (2018). *Multi-View Convolutional Neural Network Implementation in Tensorflow.* Retrieved 2019-01-05, from `https://github.com/WeiTang114/MVCNN-TensorFlow`

Luo, G., Dong, S., Wang, K., Zuo, W., Cao, S., & Zhang, H. (2018). Multi-Views Fusion CNN for Left Ventricular Volumes Estimation on Cardiac MR Images. *IEEE Transactions on Biomedical Engineering*, *65*(9), 1924-1934. doi: 10.1109/TBME.2017.2762762

Maturana, D., & Scherer, S. (2015). VoxNet: A 3D Convolutional Neural Network for Real-time Object Recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (p. 922-928). doi: 10.1109/IROS.2015.7353481

Meyers, P. H., Nice, C. M., Becker, H. C., Nettleton, W. J., Sweeney, J. W., & Meckstroth, G. R. (1964). Automated Computer Analysis of Radiographic Images. *Radiology*, *83*(6), 1029-1034. doi: 10.1148/83.6.1029

Mortazi, A., Karim, R., Rhode, K. S., Burt, J., & Bagci, U. (2017). CardiacNET: Segmentation of Left Atrium and Proximal Pulmonary Veins from MRI Using Multi-View CNN. *Computing Research Repository*, *abs/1705.06333*. Retrieved from `http://arxiv.org/abs/1705.06333`

Muhic, E. (2016). *Knee patella.* Retrieved 2019-01-12, from `https://www.physio-pedia.com/File:Knee-patella.jpg`

Olivier Moindrot, G. G. (2018). *CS230 Deep Learning.* Retrieved 2019-01-05, from `https://cs230-stanford.github.io/train-dev-test-split.html`

Osteoarthritis Initiative. (2019). *Osteoarthritis Initiative Website.* Retrieved 2019-01-28, from `https://data-archive.nimh.nih.gov/oai/`

Princeton University. (2019). *Princeton ModelNet.* Retrieved 2019-01-26, from `http://modelnet.cs.princeton.edu/`

Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., & Guibas, L. J. (2016). Volumetric and Multi-view CNNs for Object Classification on 3D Data. *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 5648-5656. doi: 10.1109/CVPR.2016.609

Research Group Therapy Planning at Zuse Institute Berlin. (2019). *Research Group Therapy Planning.* Retrieved 2019-01-05, from `www.zib.de/visual/therapy-planning`

rlczddl, & Potuaud, S. (2017). *Issue#14: Class predicted always 0.* Retrieved 2019-01-13, from `https://github.com/WeiTang114/MVCNN-TensorFlow/issues/14`

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, *115*(3), 211-252. doi: 10.1007/s11263-015-0816-y

Sebag, M. (2014). A tour of Machine Learning: An AI perspective. *AI Communications*, *27*(1), 11-23. Retrieved from `https://hal.inria.fr/hal-01109768` doi: 10.3233/AIC-130580

Sheng, B. K. Y. (2018). *VoxNet TensorFlow Implementation.* Retrieved 2019-01-05, from `https://github.com/Vectorized/VoxNet-Tensorflow`

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., ... Hassabis, D. (2016). Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, *529*, 484-489. doi: 10.1038/nature16961

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *Computing Research Repository*, *abs/1409.1556*. Retrieved from `http://arxiv.org/abs/1409.1556` doi: 10.1.1.740.6937

Singh, P., & Pearlman, S. (2017). Use of Computer Imaging in Rhinoplasty: A Survey of the Practices of Facial Plastic Surgeons. *Aesthetic Plastic Surgery*, *41*(4), 898-904. doi: 10.1007/s00266-017-0858-3

Stalling, D., Westerhoff, M., & Hege, H.-C. (2005). 38 - amira: A Highly Interactive System for Visual Data Analysis. In *Visualization Handbook* (p. 749-767). Burlington: Butterworth-Heinemann. Retrieved from `http://www.sciencedirect.com/science/article/pii/B978012387582250040X` doi: 10.1016/B978-012387582-2/50040-X

Su, H. (2019). *Readme of Original Multi-View Convolutional Neural Network Implementation.* Retrieved 2019-01-13, from `https://github.com/suhangpro/mvcnn/blob/master/README.md`

Su, H., Maji, S., Kalogerakis, E., & Learned-Miller, E. G. (2015). Multi-view Convolutional Neural Networks for 3d Shape Recognition. In *Proceedings of the 2015 IEEE International Conference on Computer Vision* (p. 945-953). IEEE Computer Society. doi: 10.1109/ICCV.2015.114

Suzuki, K. (2017). Overview of Deep Learning in Medical Imaging. *Radiological Physics and Technology*, *10*(3), 257-273. doi: 10.1007/s12194-017-0406-5

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2015). Going Deeper with Convolutions. In *Computer Vision and Pattern Recognition.* Retrieved from `http://arxiv.org/abs/1409.4842` doi: 10.1109/CVPR.2015.7298594

Tack, A., Mukhopadhyay, A., & Zachow, S. (2018). Knee Menisci Segmentation using Convolutional Neural Networks: Data from the Osteoarthritis Initiative. *Osteoarthritis and Cartilage*, *26*(5), 680-688. doi: 10.1016/j.joca.2018.02.907

Tajbakhsh, N., Shin, J. Y., Gurudu, S. R., Hurst, R. T., Kendall, C. B., Gotway, M. B., & Liang, J. (2016). Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? *IEEE Transactions on Medical Imaging*, *35*, 1299-1312. doi: 10.1109/TMI.2016.2535302

Tanna, S. (2013). *Priority Medicines for Europe and the World, Background Paper 6.12 Osteoarthritis.* Retrieved 2019-01-28, from `https://www.who.int/medicines/areas/priority_medicines/BP6_12Osteo.pdf`

Toennies, K. D. (2012). *Guide to Medical Image Analysis Methods and Algorithms.* Springer Verlag.

Universitätsbibliothek der Freien Universität Berlin. (2019). *Bibliotheksportal Primo der Freien Universität Berlin.* Retrieved 2019-01-05, from `http://primo.fu-berlin.de/`

Universitätsbibliothek der Humboldt Universität Berlin. (2019). *Bibliotheksportal Primus der Humboldt Universität Berlin.* Retrieved 2019-01-05, from `https://primus.ub.hu-berlin.de/`

Universitätsbibliothek der Technischen Universität Berlin. (2019). *Bibliotheksportal Primo der Technischen Universität Berlin.* Retrieved 2019-01-05, from `https://portal.ub.tu-berlin.de/`

US National Library of Medicine at the National Institutes of Health. (2019). *PubMed Search Engine.* Retrieved 2019-01-05, from `https://www.ncbi.nlm.nih.gov/pubmed/`

Wallach, I., Dzamba, M., & Heifets, A. (2015). AtomNet: A Deep Convolutional Neural Network for Bioactivity Prediction in Structure-based Drug Discovery. *Computing Research Repository*, *abs/1510.02855*. Retrieved from `http://arxiv.org/abs/1510.02855`

Wang, L., Duan, X., Zhang, Q., Niu, Z., Hua, G., & Zheng, N. (2018). Segment-Tube: Spatio-Temporal Action Localization in Untrimmed Videos with Per-Frame Segmentation. In *Sensors.* doi: 10.3390/s18051657

Wells, W. M. I. (2016). Medical Image Analysis - past, present, and future. *Medical Image Analysis*, *33*, 4-6. doi: 10.1016/j.media.2016.06.013

World Health Organization. (2012). *Priority Medicines for Europe and the World: A Public Health Report To Intervention.* Retrieved 2019-01-28, from `http://www.eumusc.net/myUploadData/files/Musculoskeletal%20Health%20in%20Europe%20Report%20v5.pdf`

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3D ShapeNets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition* (p. 1912-1920). doi: 10.1109/CVPR.2015.7298801

Zhu, W., Liu, C., Fan, W., & Xie, X. (2017). DeepLung: 3D Deep Convolutional Nets for Automated Pulmonary Nodule Detection and Classification. *Computing Research Repository*, *abs/1709.05538*. Retrieved from `http://arxiv.org/abs/1709.05538`

Zilles, K., & Tillmann, B. (2010). *Anatomie*. Berlin, Heidelberg: Springer Berlin Heidelberg.

DVD Attachment

Please find the file with MRI scan IDs, the code used to load and preprocess the data, to train and evaluate the networks as well as the trained CNNs on this DVD. See Readme.txt in the root directory.

The MRI scans cannot be provided here due to reasons of copyright. They can, however, be obtained from https://data-archive.nimh.nih.gov/oai/.

Freie Universität Berlin

Selbstständigkeitserklärung

Name:	(Nur Block- oder Maschinenschrift verwenden.)
Vorname:	
geb.am:	
Matr.Nr.:	

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende _____ selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch bei keiner anderen Universität als Prüfungsleistung eingereicht.

Datum: _____ Unterschrift: _____