Eirik Gromholt Homlong

Computer-Aided Diagnostics: Segmentation of Knee Joint Anatomy Using Deep Learning Techniques

Master's thesis in Simulation and Visualization
Supervisor: Ibrahim A. Hameed Robin Trulssen Byeand Webjørn Rekdalsbakken
July 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

NTNU
Norwegian University of
Science and Technology

Abstract

Segmentation is the process of dividing images into multiple meaningful parts and regions with similar attributes. Image segmentation is heavily utilized in medical imaging and allows doctors to gain additional diagnostic insight. Semi-automatic segmentation tools are available today, but still, require a lot of user input and is highly time-consuming.

In recent years there has been a rapid advance in machine learning techniques for computer vision applications, which have also proven to be useful for medical image segmentation. Using a 3D implementation of the fully convolutional neural network U-Net model we have implemented a fully automated process for semantic segmentation of the bones, the anterior cruciate ligament (ACL) and posterior cruciate ligament (PCL) of the knee joint. We find that the model is able to segment all three components accurately.

A platform for creating and testing different model pipelines has also been developed, including a graphical user interface (GUI) to visually compare the predicted segmentation masks to their ground truth counterparts. The platform is lightweight and flexible, and can easily be adapted to other segmentation tasks in the future.

Sammendrag

Segmentering er prosessen i å dele bilder inn i flere meningsfulle deler og regioner med lignende egenskaper. Bildesegmentering er sterkt utnyttet i medisinsk bildebehandling, og gjør det mulig for leger å få ekstra diagnostisk innsikt. Semiautomatiske segmenteringsverktøy er tilgjengelig i dag, men krever fortsatt mye manuelt arbeid og er svært tidkrevende.

I de siste årene er det blitt gjort store fremskritt i maskinlæringsteknikker for bildedata, som også har vist seg å være nyttig for medisinsk bildesegmentering. Ved hjelp av en 3D-implementering av det fully convolutionalnevrale nettverket U-Net har vi implementert en helautomatisk prosess for semantisk segmentering av bein, fremre korsbåndet (ACL) og bakre korsbåndet (PCL). Vi finner at modellen er i stand til å nøyaktig segmentere alle tre komponentene.

En plattform for å lage og teste forskjellige modellkonfigurasjoner er også utviklet, inkludert et grafisk brukergrensesnitt (GUI) for visuelt å sammenligne de predikerte segmenteringsmasker mot de reelle (ground truth) segmenteringsmaskene. Plattformen er lett og fleksibel, og kan lett tilpasses til andre segmenteringsoppgaver i fremtiden.

Preface

This Master's thesis is the final deliverable work of the Simulation and Visualization Master's program at the Department of ICT and Engineering at Norwegian University of Science and Technology (NTNU) in Ålesund. The work presented in this thesis was carried out through the spring semester of 2019.

The thesis concerns the automatic segmentation of magnetic resonance images of knee joints. The motivation for choosing this thesis comes from a personal interest in machine learning, computer vision and visualization. And this thesis should be covering all of these fields.

Finally, I would like to thank my supervisors Ibrahim A. Hameed, Robin Trulssen Bye and Webjørn Rekdalsbakken for guidance and feedback throughout the thesis. I would also like to thank Sunnmøre MR-klinikk AS for providing the data for this thesis, and also further thank Kjell-Inge Gjesdal from the clinic for helping with the thesis and taking time to explain details regarding the data provided. I want to thank my friend Jan-Eirik Welle Skaar, my brother Olav Aleksander Homlong, and my sister Maria Kristin Homlong, for helping me with the thesis. I would also like to thank the rest of my family and friends for all their support throughout my years studying in Ålesund.

Contents

List of Figures

List of Tables

Abbreviations

2D two-dimensional. 10, 19, 46, 70, 73, 75, 87, 96, 104, 105, 124, 128, 133, 134, 149

3D three-dimensional. 10–12, 19–22, 46, 58, 65, 70, 73, 75, 87, 96–98, 100–103, 105, 106, 124, 126, 128–130, 133, 134, 149–152

AAM active appearance model. 31

ACL anterior cruciate ligament. 11, 12, 23, 32, 80, 88, 103, 115, 117, 119–122, 130, 135, 149, 152

AI artificial intelligence. 34

ANN artificial neural network. 23, 34, 36, 38–40, 42, 43, 47, 48, 55

AR augmented reality. 19

ASM active shape model. 31

BAGGING bootstrap aggregation. 56

BGD batch gradient descent. 36, 37

BoM bucket of models. 56

BraTS Brain Tumor Segmentation. 75

CED convolutional encoder-decoder. 10, 73, 74

CNN convolutional neural network. 47, 48, 50, 59, 60, 62, 64, 67, 90, 136

CRF conditional random field. 57, 59, 77, 78

CT computer tomography. 25, 70

DCRF Dense conditional random field. 11, 13, 73, 77, 78, 87, 91, 93, 98, 104, 117–119, 130, 131, 134

DNN deep neural network. 47

DSP digital signal processing. 44

FCN fully convolutional network. 31, 64, 65, 67, 73, 77, 86

fMRI functional magnetic resonance imaging. 26

FN false negative. 54

FNR false negative rate. 77

FP false positive. 54

FPR false positive rate. 77

FS fat suppressed. 8, 12, 27, 29, 80, 85, 111, 115, 120, 123, 154

FSE fast spin echo. 74

GA genetic algorithm. 56

GDL generalized dice loss. 75, 89, 101, 108, 129

GPU graphical processing unit. 60, 99, 101, 136

GUI graphical user interface. 79, 96, 98, 104, 133

ILSVRC14 ImageNet Large Scale Visual Recognition Challenge 2014. 59, 61, 62

ILSVRC15 ImageNet Large Scale Visual Recognition Challenge 2015. 62

INI initialization. 90

IoU intersection over union. 53, 70, 72

JPEG joint photographic experts group. 96

JSON JavaScript Object Notation. 90

LCL lateral colleteral ligament. 24

MAP maximum a posteriori. 87

MBGD mini-batch gradient descent. 37

MCL medial colleteral ligament. 24

ML machine learning. 20, 21, 23, 30, 34, 43, 44, 55, 59, 61, 79, 81, 87, 93

MRI magnetic resonance imaging. 8, 13, 18–23, 25–29, 31, 32, 44, 58, 65, 73, 74, 76, 77, 79–81, 83, 90, 91, 96, 98, 100, 102, 111, 127–134

MRS magnetic resonance spectroscopy. 26

MSE mean squared error. 35, 36

NMR nuclear magnetic resonance. 26

PAM probabilistic atlas models. 31

PCL posterior cruciate ligament. 11, 12, 23, 32, 80, 88, 103, 112, 115, 117, 119, 120, 122, 130, 135, 149, 151

PD proton density weighted. 8, 12, 27, 28, 80, 85, 111, 120, 123, 153, 155

PET positron emission tomography. 25, 76, 77

PNG portable network graphics. 96

PSO particle swarm optimization. 56

PTL patellar tendon ligament. 24

ReLU rectified linear unit. 42, 60

ResNet residual neural network. 62, 63

RNN recurrent neural network. 40

SGD stochastic gradient descent. 37

SS Sensitivity - Specifity. 75

SSM statistical shape model. 31

STD standard deviation. 88, 107

T1 T1 weighted. 8, 12, 27, 28, 80, 85, 100, 101, 111, 115, 117, 120, 123, 130, 131, 153, 156

T2 T2 weighted. 28

TanH hyperbolic tangent. 41

TN true negative. 54

TP true positive. 54

URL uniform resource locator. 86, 91, 92

VGG Visual Geometry Group. 9, 61, 73

VGGNet VGG Network. 61, 62

VR virtual reality. 19

VTK visualization toolkit. 98

WCCL weighted categorical crossentropy loss. 88, 89, 101–103, 108, 109, 128

WCE weighted cross-entropy. 75, 108

WDL weighted Dice loss. 88, 89, 101–103, 108, 109, 128, 129

WJL weighted Jaccard loss. 88, 89, 101, 108, 128

YAML YAML Ain't Markup Language. 90

Chapter 1: Introduction

This chapter presents an introduction to this master thesis. Section 1.1 introduces the MRI segmentation problem and discusses the importance of it, as well as the motivation behind, solving the problem. In section 1.2, the scope of the thesis and the problem will be further explored. Section 1.3 will give a description of the goals of the research and research questions for this master thesis.

1.1 Background and motivation

Computer vision is a field that has greatly expanded with the recent development of machine learning techniques [61]. Computer vision is useful for a wide range of tasks involving finding information in images and allows for a better understanding of images. A significant portion of diagnosing in the medical field is done using medical imaging and requires a lot of manual labour. Because of this, the use of computer vision algorithms for medical imaging seems beneficial and can be a helpful tool to automate parts of the diagnostic processes.

Radiology is a medical speciality where radiological imaging is used to diagnose and treat diseases. One common technique in radiology is MRI. MRI is a medical imaging technique used in medical applications to take images of the underlying physiological processes of a patients body and anatomy. MRI images are primarily used to diagnose the patients by having a radiologist analyze them. MRI segmentation is useful for many medical applications. At the moment, most radiologists manually segment the different anatomical structures in the MRI images. Manual segmentation is a tedious task which involves the radiologists segmenting it by hand, which can be both physically and mentally straining, consuming a lot of time which could otherwise be better allocated.

Norges teknisk-naturvitenskapelige universitet (NTNU), Sunnmøre MR-klinikk and Ålesund Sjukehus are planning a more extensive collaboration project for Computer-Aided Diagnostics. The proposed project has the end goal of establishing automatic 3D segmented models with diagnostic information of human joints, and the ability to interact with these models using virtual reality (VR)/augmented reality (AR), cinematic rendering visualization and physical and virtual surgical tools. This master thesis focuses mainly on the MRI segmentation part of the project, as it is one of the critical components that must be established before the other desired features can be implemented. The segmentation part of the project will primarily be a collaborative effort with Sunnmøre MR-klinikk.

1.2 Problem and scope

Image segmentation of MRI images is a complex problem with a lot of variables. The MRI images are layered, making it hard to find the regions of interest. Individual MRI images also contain a lot of noise from tissue-related, motion-related and technique-related artefacts. The images are monochrome, which means there is less information in the images, having only one colour channel per pixel, compared to coloured images, which usually have three channels of colour information. The resolution of the images is also something to consider. Further there also exist different types of MRI images and weightings for MRI images. For example, some images are better for resolving the fluids in the body. Another problem is how the images are represented on the sagittal, coronal and axial planes. The planes consist of several slices that give 3D volume made up of voxels. Working with 3D volumes is a more complex problem than segmenting on single 2D images, and it will be challenging to find the correct approach to these issues.

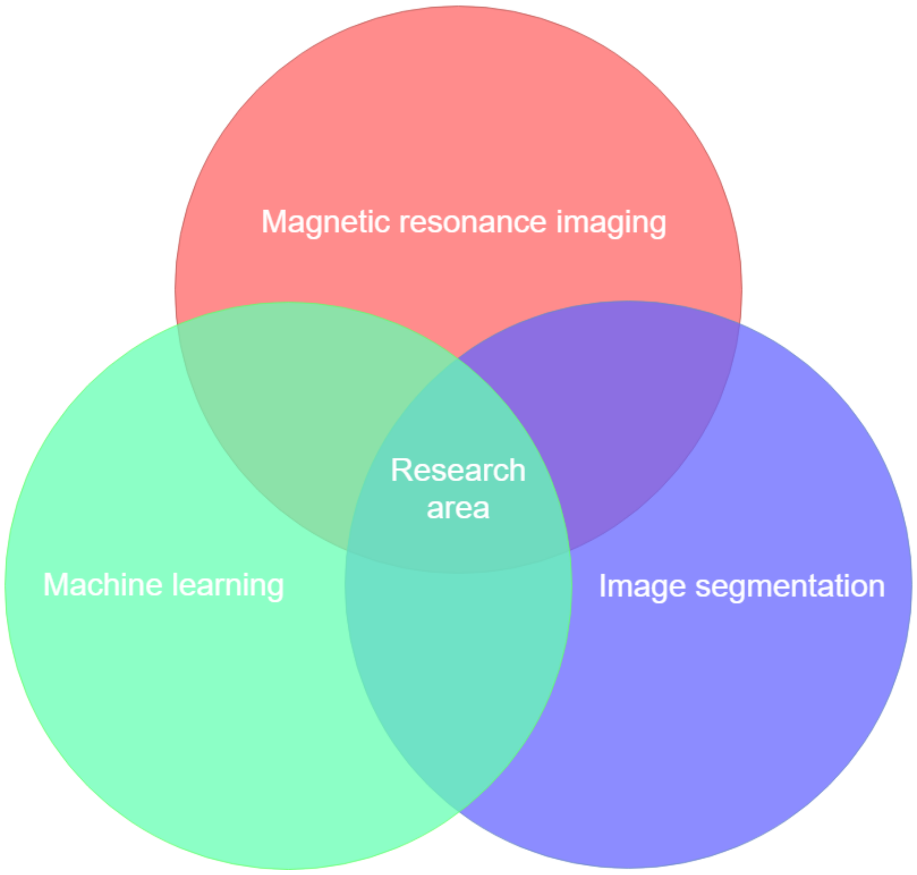


Figure 1.1: Scope of the thesis

The scope of this thesis will further be defined to be segmentation of anatomical structures of interest in MRI images using machine learning (ML) techniques. The segmented images will mainly be used for 3D reconstruction of the segmented anatomical features to be displayed in a virtual environment. It is desired that the 3D reconstruction can be used as an aiding tool for diagnosing patients. Thus it is crucial to segment the desired anatomical features with sufficient accuracy. For this thesis, the main focus was to establish methods and tools for segmentation of the knee joint.

The knee is a complex joint which consists of multiple components. It could be a hard challenge to make the automated segmentation system distinguish these components, as some of them are very similar, others hard to detect, and there are also proximity issues to take into consideration. Another problem is how the different components vary in size as ML techniques often develop biases when training on imbalanced data. And as every knee is different, the components also will vary in location, size, rotation and also may include various anatomical anomalies.

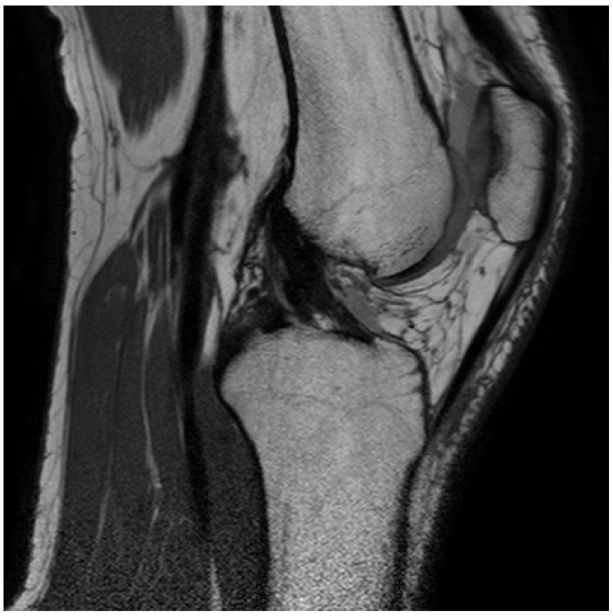


Figure 1.2: Example MRI image of knee joint.

1.3 Research goals and questions

The main goal for this thesis is to develop methods and tools for automatically segmenting anatomical features of interest in MRI images using ML algorithms, and make it useful for real-world applications. Since it is important that the system is accurate, it will be evaluated using segmentation accuracy metrics, and also by inspection. The system should also be used for generating 3D models for a 3D simulated environment. Furthermore, three following research questions will be proposed for this thesis:

Research question 1: *Does the proposed tools and methods yield accurate segmentation?*

The accuracy of the segmentation is typically determined by evaluation metrics such as pixel accuracy and overlap measures. To compare the segmentation methods, usually, a "ground truth" or a gold standard segmentation mask is needed for comparison. However, as the manual segmentation of the ground truth mask is done by a human expert, it is prone to errors, and the expert's interpretation of the images are highly subjective [18]. When considering this, it is essential that the final segmentation masks are also evaluated by inspection as we can't rely on the numerical evaluation metrics alone.

Research question 2: *Is the proposed tools and methods adaptable and will it be applicable for different segmentation tasks for MRI images?*

Because the collaboration project will likely expand to include multiple different anatomical structures of the body, the proposed tools and methods should preferably not only be able to solve one specific segmentation task but be usable for a wide range of segmentation problems regarding MRI. When given a new segmentation task, the proposed methods and tools should be adaptable. A further segmentation should preferably be solved by only changing the parameters, or apply minor changes to the methods and tools.

Research question 3: *Is the tools and methods ready for real-world applications?*

As the segmentation masks are going to be used to create 3D models for a simulator, it would be desirable that the proposed tools and methods are ready to be used for real-world applications. Since the tools and methods may be used by doctors and radiologists, the tools and methods should preferably be usable and adaptable without having to change much of the code. Best case scenario the tools and methods should be compiled into executable programs ready to be used for the intended application.

1.4 Thesis structure

The thesis is divided into seven chapters. Chapter 1 addresses the motivation behind the thesis, explain the scope and problem of the thesis and describes the research questions for the thesis. Chapter 2 explains the necessary background theory for the thesis. Chapter 3 presents literature and research for the thesis. Chapter 4 presents and discusses the methods used to obtain the results of the thesis. Chapter 5 explains the different experiments conducted using the methods. Chapter 6 presents the results of the experiments and the general results of the thesis. Chapter 7 discusses the results of the thesis. Chapter 8 concludes the thesis and proposes future work.

Chapter 2: Background theory

This chapter presents the background theory on knee anatomy, MRI, ML and segmentation. The chapter will briefly go over theory of knee anatomy and MRI, but will also go deeper into the theory regarding segmentation and machine learning. Section 2.1 will briefly explain the anatomy of the knee joint. Section 2.2 will explore different MRI techniques and further discuss different types of MRI images and different file formats. Section 2.4 will go through different computer vision methods focusing on segmentation and further explain how semantic segmentation works. Section 2.5 will explain artificial neural network (ANN)s and different ML paradigms regarding them. In section 2.6 the convolution operation will be discussed, and in section 2.7 convolutional neural networks will be further discussed. Section 2.8 will explain auto encoders. Section 2.9 will go through different methods to numerically evaluate semantic segmentation. In section 2.12 parameter searches for machine learning methods will be explained. And finally in section 2.14 the marching cubes algorithm will be briefly explained.

2.1 Anatomy of the knee

The knee is one of the most complex joints in the human body. The knee consists of two joints functioning as a hinge joint. The knee also consists of several bones including the thigh bone, shin bone, calf bone and the knee cap, which are also referred to by their Latin names femur, tibia, fibula, and patella. The cartilage between the femur and tibia, also known as the medial and lateral menisci, provides shock absorption. The knee also consists of ligaments providing stability for the knee. The ACL prevents the femur from sliding back towards the tibia. Moreover, the PCL prevents the femur from sliding towards the tibia. patellar tendon ligament (PTL) holds the patella in place. The medial colleteral ligament (MCL) and lateral colleteral ligament (LCL) prevent sideways motion of the femur. Movement of the knee joint is controlled by the muscles surrounding it. [30]

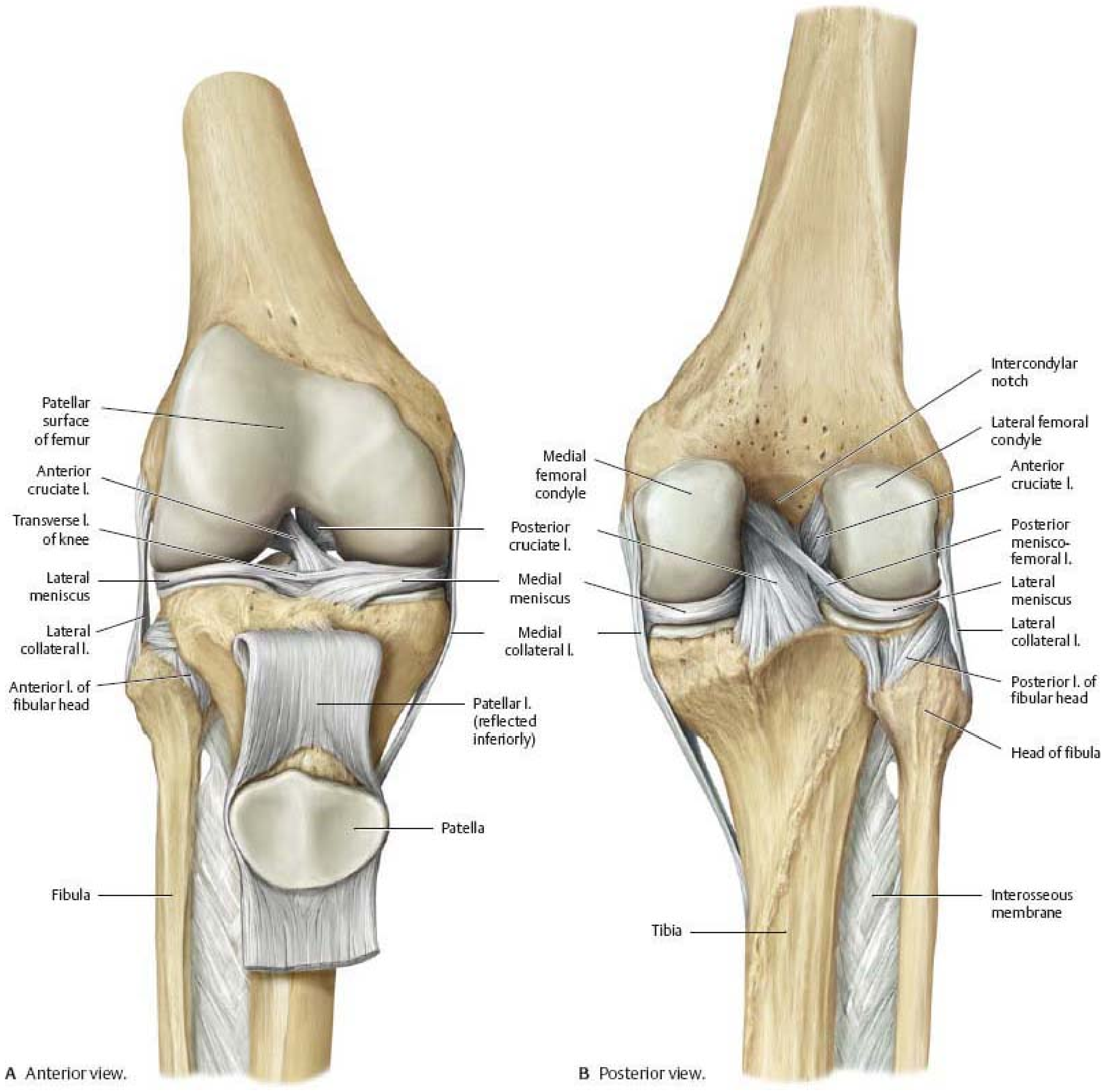


Figure 2.1: Anatomy of the knee [59]

2.2 Magnetic resonance imaging

Radiology is, as mentioned earlier, a medical field using different techniques of imaging to reveal and treat various conditions in animals and humans [86]. Among these types of imaging techniques we have computer tomography (CT), ultrasound, positron emission tomography (PET) and MRI. This thesis will focus on MRI.

An MRI scanner uses magnetic fields, magnetic field gradients, and radio waves to produce detailed images of organs inside the bodies of humans or animals [73]. Most of the human body is made up of water molecules, which consist of hydrogen and oxygen atoms. At the centre of each hydrogen atom is a particle called proton (some of them have added neutrons, but these nuclei are not detected in MRI) [78]. The magnets in the scanner have two different tasks; the first one is to make the spin of the protons align in the same direction. Another magnet is turned on and off in quick pulses making the hydrogen atoms change their alignment when the magnet is on. Then when it switched off, they gradually revert to the equilibrium configuration as previously. In this relaxation process, the protons in the cells emit radio waves that are picked up by receivers [78]. The contrast between the different tissues is determined by the rate at which excited nuclei return to the relaxed state [73].

Nuclei can have an intrinsic spin-dependent on its internal configuration. If non-zero, this spin makes each nucleus behave like a magnet. The spin states have different energies depending on the alignment with the magnetic field and its local field strength. Parallel alignments have lower energy levels, while antiparallel alignments have slightly higher energy levels. Because the distribution of the spin alignments at thermal equilibrium follows a Boltzmann distribution, the relaxation from roughly equally distributed alignments produces a clear radio signal that can be detected. This relaxation follows an exponential function with a relaxation constant as a factor in the exponent. [63, 32].

2.2.1 MRI techniques

There are several different MRI techniques. Some of them are listed as follows:

- nuclear magnetic resonance (NMR). This technique is used for structural analysis of molecules in physics and chemistry. [13][99]
- MRI. This technique used in medicine to take detailed pictures inside the body. The pictures are sectioned images of the part of the body the doctors want to examine. MRI is the medical application of NMR. [13][99]
- magnetic resonance spectroscopy (MRS). This technique is often used to measure different substances in human tissue. [13][99]
- functional magnetic resonance imaging (fMRI). This technique is used to measure functions and changes in the blood flow in the brain. [31] [13][99]

2.3 MRI images

The images the MRI scanner produces show tissue inside the body with great detail. The images are captured in the axial, coronal, and sagittal plane. Image data is stored as images through layers. The individual pixels in the layers are called voxels. The voxels have width, height, and depth dimensions, unlike pixels which only have the width and height dimensions. MRI images are usually referred to as volumes as they are built up of voxels.

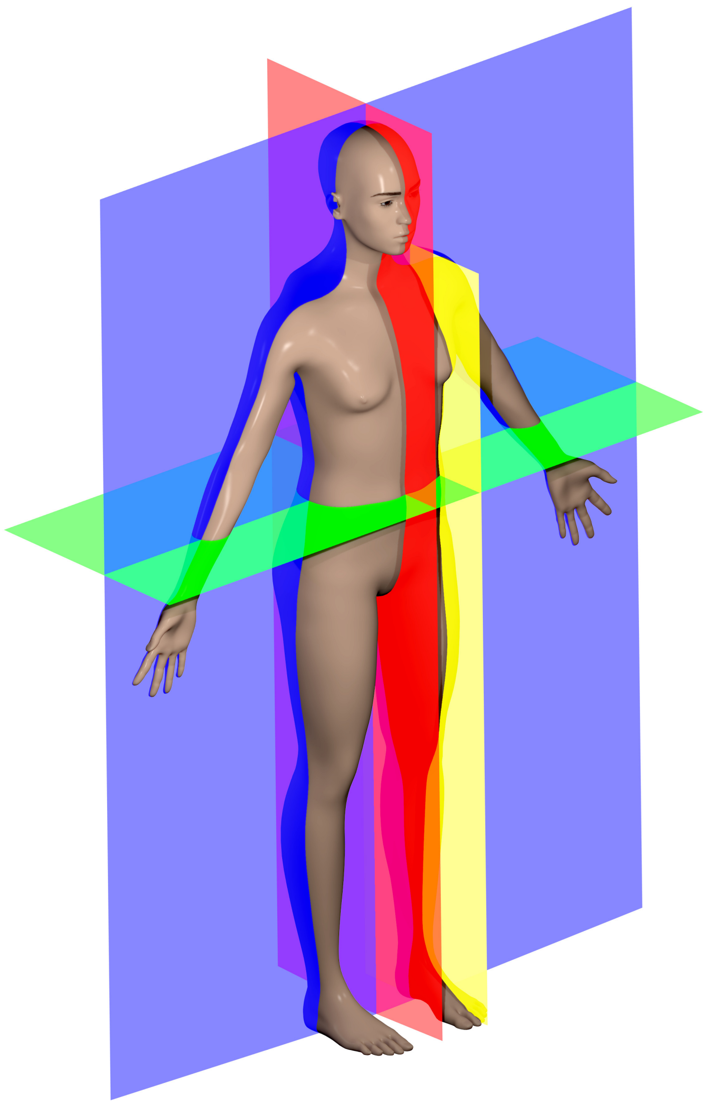


Figure 2.2: Human anatomy planes. The red plane is the sagittal plane, yellow the parasagittal plane, blue the coronal plane and green the axial plane. [44]

2.3.1 MRI image types

There are several MRI image types, and they differ in the way they are weighted. The weighting affects which tissues are visible and how the tissues appear in the image. The different weightings are useful depending on which tissues need examination. This thesis will focus on T1, PD and FS images as these were the types of images provided.

T1 weighted images

A T1 weighted image is a type of MRI image which presents the differences in the T1 relaxation times of the anatomical tissues. Fat appears bright in the T1 weighted images, and water appears darker. [108, 76]

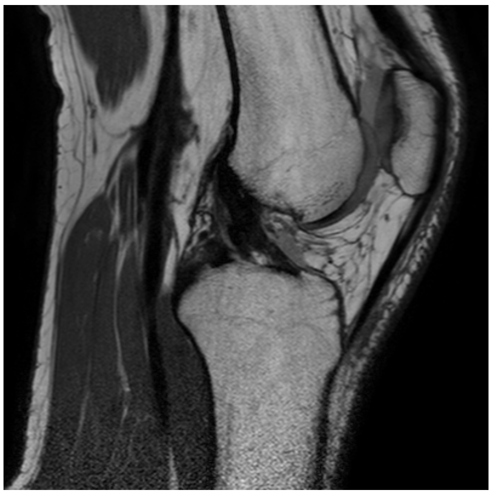


Figure 2.3: T1 weighted image example

PD weighted images

PD images are the result of a minimization of the T1 and T2 weighted (T2) contrasts. Fat tissue gives strong signals and appears bright on the PD images. Fluids emit medium signals and appear as intermediate brightness. [11]

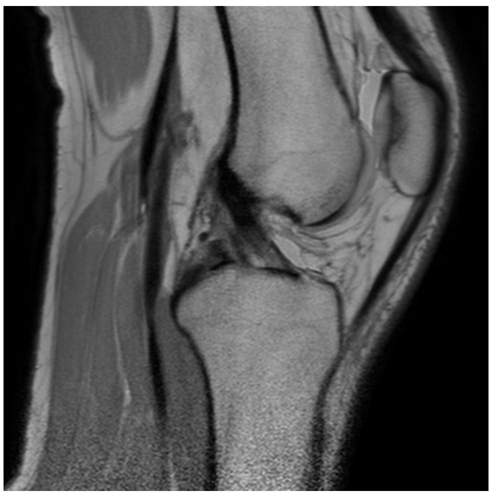


Figure 2.4: PD weighted image example

Fat suppressed images

FS is a commonly used method for MRI images. The method suppresses the signal from adipose tissue. However, the method is also used in some cases to detect adipose tissue. Because of the suppression, the fat tissue appears darker, and in contrast, the fluids will appear brighter in theFS images. [25]

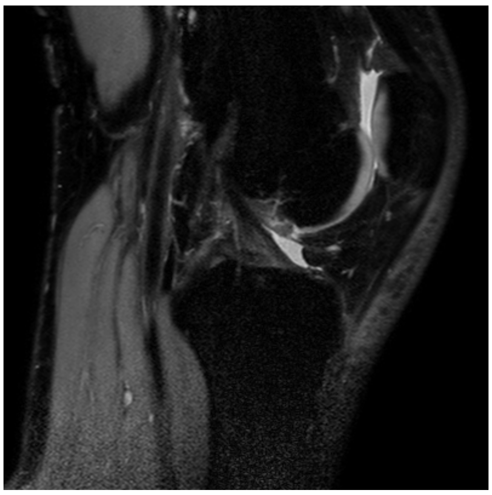


Figure 2.5: FS image example

2.3.2 Image formats

MRI data come in several formats. The main formats are Analyze, Minc, Dicom, and Nifti [64]. Since the data provided for this thesis was mostly formatted in the Nifti format, this format will be the main focus.

The Nifti format is rapidly becoming the default format for software used for medical imaging. The Nifti format can be seen as an improvement and update of the Analyze format as the Nifti format support for additional header information. Notably, the header of the Nifti format contains additional information for rotation and image orientation. The Nifti files are usually saved as a single file with the extension ".nii", but the format also allows for both storing the header data and the pixel/voxel data separately. [64]

2.4 Segmentation

Segmentation is a big field in digital image processing and is especially important for medical imaging. The process of image segmentation divides images into multiple meaningful parts and regions with similar attributes like texture, colour, and intensity [55, 7, 48]. The overall goal of segmentation is to simplify the images to make them easier to analyze. Image segmentation can be done using simple thresholding methods or state of the art ML techniques [115, 47]. Image segmentation can be used for various applications, e.g., medical, facial recognition, and autonomous driving [88, 37, 56].

Because there are no general segmentation procedures, the choice of segmentation techniques may be subjective [94]. The technique implementation and usefulness will always vary depending on the types of data one possess. Conventional techniques for image segmentation include the thresholding method, edge detection based techniques, region-based techniques, clustering-based techniques, and artificial neural network-based techniques, and more [55].

It is important to distinguish types of computer vision techniques and image segmentation terminology from one another as it helps understanding which methods to pursue given a computer vision problem. Explanation of the different methods are as follows:

- **Object recognition:** The goal of object recognition is to detect and localize all objects within a specified limited set of classes in an image. For visualization, the objects are usually highlighted at their location with a bounding box surrounding them and a label to indicate which class of object it is. [81]
- **Object detection:** The goal of object detection is similar to the goal of object recognition. The difference is that the object detection technique finds instances of objects in images compared to objects of different classes. For visualization, the objects detected are typically highlighted with a bounding box around them. [81, 80]

- **Image segmentation:** The goal of image segmentation is to take an image and split it into multiple segments consisting of pixels (superpixels). Image segmentation is usually used to find objects or boundaries within an image. [115, 47]
- **Semantic segmentation:** Semantic segmentation is similar to image segmentation. The objective semantic segmentation is to assign each pixel of the image a segment class, unlike image segmentation, where the objective is to segment unlabeled regions of interest in the image. [106, 69]
- **Instance segmentation:** The objective of instance segmentation is similar to semantic segmentation. The difference is that the goal of instance, segmentation is to outlay individual instances of one or several semantic classes in an image. [24, 112]

2.4.1 MRI segmentation methods

For the segmentation of MRI images, there is no common segmentation method that can be used for all types of MRI images. Many different segmentation methods are useful, depending on the segmentation problem. The different methods are usually categorized as model-based, image-based, or hybrid methods.

The model based methods are methods where landmark positions are determined by minimizing an energy function. When the energy function is minimized the landmarks will form a contour around the segmentation target, and can for example be used to generate a dense segmentation mask by filling the contour. Some examples of model-based methods are statistical shape model (SSM) [40], active appearance model (AAM) [21], active shape model (ASM) [36], and probabilistic atlas models (PAM) [95].

The image-based segmentation methods are methods where each voxel in the volume is labelled, and this is often referred to as dense segmentation. Some examples of image-based segmentation methods include level set [84, 65], graph cut [1, 97], and fully convolutional network (FCN) [69]. The hybrid segmentation methods are often combinations of image-based methods and model-based methods. An example of a hybrid model is the shape aware FCN [72].

2.4.2 Dense semantic segmentation

Because this thesis will focus on dense semantic segmentation methods, it is important to know how the underlying mechanics of these methods work. As mentioned earlier, dense segmentation methods are methods where each voxel in a volume is labelled. For the problem of this thesis, the goal is to take a MRI volume of dimensions $(width \times height \times depth)$ and output a segmentation mask where each voxel contains a class label represented as an integer $(width \times height \times depth \times 1)$. The semantic classes include bone, PCL, ACL and background. The background class is useful for semantic segmentation problems where parts of the image will be unlabeled.

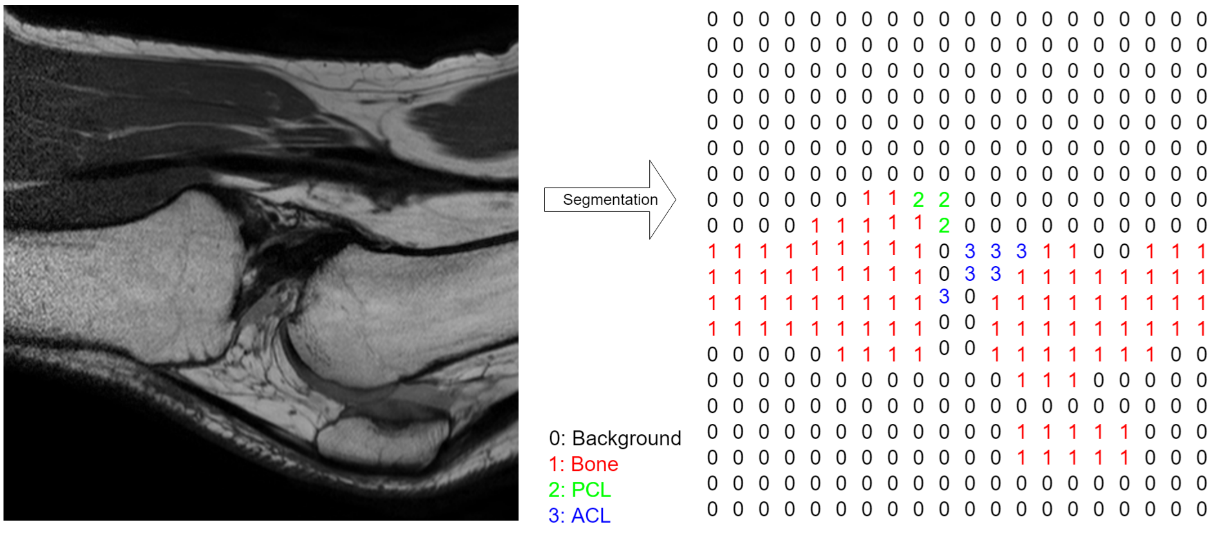


Figure 2.6: Low resolution prediction mask visualizing the segmentation mask for a slice in a MRI volume. For real world application the segmentation mask should match the input resolution.

Furthermore, for a dense segmentation method for a multi-class problem the prediction and target (ground truth) is typically represented by a one-hot encoding of segmentation map where we create an output channel for each segment class including the background class [101]. The final shape of the segmentation map is $(width \times height \times depth \times N)$ where N is the number of segment classes. The prediction is usually the product of an inference model, e.g., a machine learning model where each class channel represents the probability or confidence of that channel being the right class. This prediction segmentation map is often referred to as a confidence map, as each encoding is holding the confidence for a specific segment class. Finally, the segmentation map can be collapsed into a segmentation mask by using the argmax function of each depth-wise voxel vector. Argmax is a function returning the index of the highest value in a vector or matrix. By using argmax, we assign the voxel with the segment class having the highest probability/confidence.

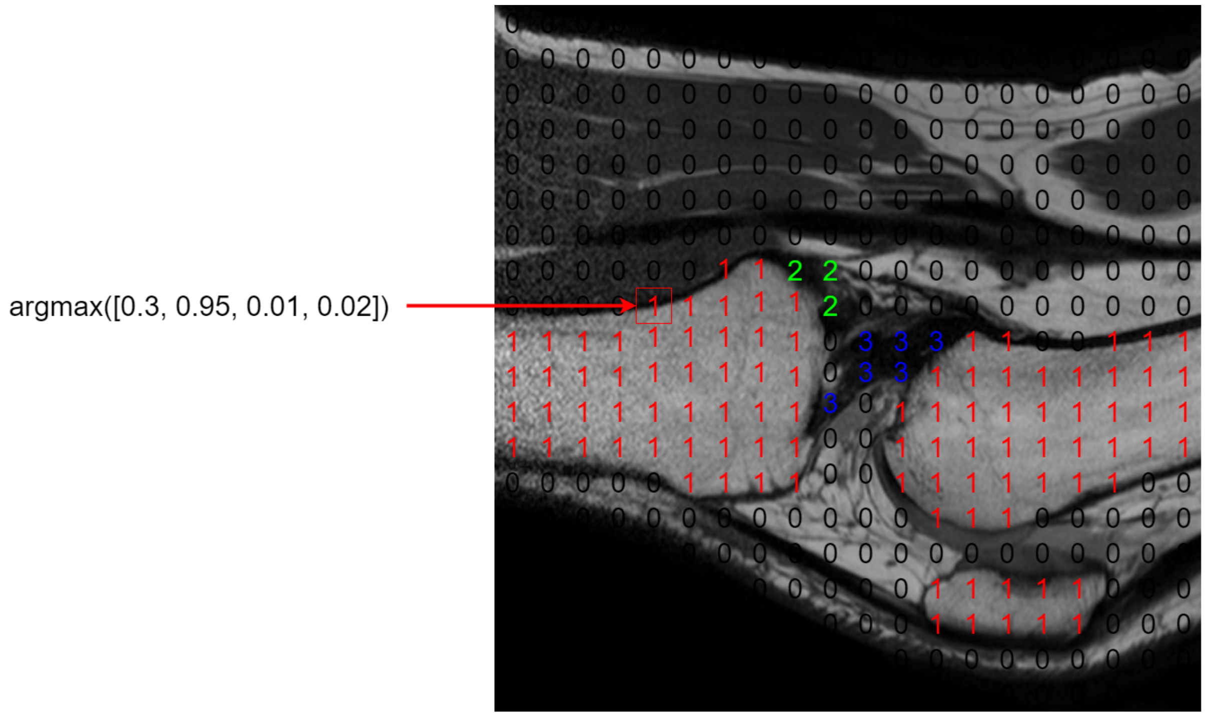


Figure 2.7: Example of how the argmax function decide the segmentation masks voxel value for a single voxel.

Another way of performing dense multi-class semantic segmentation is by setting the number for output channels to the number of classes excluding the background class. Then each class of the one-hot encoding is represented as a binary classification, and the segmentation masks classes are decided by thresholds set for each of the channels (usually set to 0.5). If none of the classes exceeds the threshold, the semantic label is set to background. If several classes exceed the threshold, the class with the highest value is set as the semantic label. [104]

2.5 Artificial neural networks

ANN is a field in artificial intelligence (AI) and ML and is a common term for a variety of different algorithms of neural networks. ANNs are inspired by neuroscience and how the neurons in the brain work. The ANN is composed of nodes or units which are connected by directed links. These nodes mimic the neurons in a brain and are usually called neurons or artificial neurons. The purpose of the neurons is to propagate values forward to other neurons when activated.

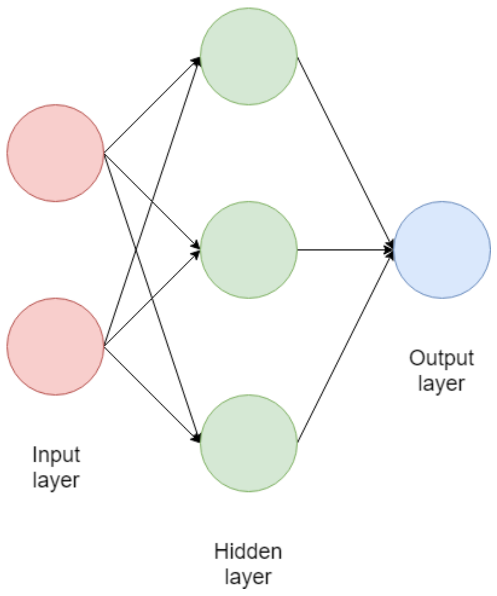


Figure 2.8: Neural network

Stuart et al. [92, p. 728] describes the neural network structure with notation and formula as follows: The neural network is composed of neurons connected by directed links. A link from neuron i to neuron j is connected to propagate activation a_i from i to j. Each neuron has an input a_i with an associated weight $w_{i,j}$. The weights are numeric values that determine the strength of the connection between neurons. Equation 2.1 presents how the neurons compute a weighted sum of their inputs. Equation 2.2 presents a function applied to this weighted sum to derive the output, also known as an activation function which will be further discussed in section 2.5.3.

$$in_j = \sum_{i=0}^{n} w_{i,j} a_i \tag{2.1}$$

$$a_j = g(in_j) = g\left(\sum_{i=0}^{n} w_{i,j}a_i\right) \tag{2.2}$$

Typically a bias value b is added for each neuron in network structure. The role of the bias is to shift the activation function by adding a constant value to the input. The bias term added to equation 2.1 gives the following equation.

$$in_j = \sum_{i=0}^{n} w_{i,j}a_i + b_j \tag{2.3}$$

2.5.1 Loss function

In the previous section, we described how the weights w determine the strength of the connections between neurons and how it affects the output of the network. To reduce the output error of the network the weights w of the network have to be adjusted. Commonly a loss function is used to measure how accurate or inaccurate the output of the network is compared to the ground truth data. The loss function provides a numerical output of how well the network performs. By using the loss function, we can determine how much each weight contributes to the overall error, which will be further discussed in section 2.5.2

A common loss function used for classification problems is the cross-entropy loss function. The cross-entropy function is given by equation 2.4. Where t is the ground truth encoding, and s is the prediction score for each training sample $_i$. E is the error, C is the number of classes and W the weight matrix of the neural network. The cross-entropy loss function measures the similarity between two probability distributions, and the cross-entropy loss decreases as the predicted probability converges to the ground truth label.

$$E(W) = -\frac{1}{C}\sum_{i}^{C} t_i log(s_i) \tag{2.4}$$

Another common loss function is the mean squared error (MSE) loss function. The MSE loss function is rarely used for classification problems but is typically used for regression evaluation. However, for a classification problem, the MSE function measures the average of the squares of the errors between the prediction and ground truth labels for each class. The MSE loss function is given by the following equation.

$$E(W) = \frac{1}{C}\sum_{i=1}^{C} \|t_i, p_i\|^2 \tag{2.5}$$

2.5.2 Gradient descent

The objective of training a ANN is to minimize the loss output from the loss function $E(W)$ by altering the weights of the network. The training is commonly done using an iterative optimization algorithm called gradient descent. Ruder et al. [89] explain the gradient descent algorithm eloquently: The basic idea of the gradient descent algorithm is to minimize an objective function $J(\theta)$ by updating the parameters θ in the opposite direction of the gradient of the objective function $\eta_\theta J(\theta)$. Equation 2.6 shows the batch gradient descent (BGD) also known as vanilla gradient descent, where η is a learning rate which determines how many steps to take towards the minimum.

$$\theta = \theta - \eta \nabla_\theta J(\theta) \tag{2.6}$$

A negative aspect of BGD is the need to calculate the gradients for the whole dataset for every single update of the parameters. Because of this BGD, can be very slow and is also intractable for some problems where the data is larger than the available memory. Another consequent disadvantage with BGD is how it is not possible to update the model on the fly with new data.

To deal with the issues of BGD, the stochastic gradient descent (SGD) algorithm can be used, it is given by equation 2.7. SGD updates each parameter for each training example x^i and label y^i, and deals with the redundancy issue of BGD, as it performs one update at a time. The SGD algorithm will usually make the training faster and can be used with new data on the fly. The disadvantage of SGD algorithm is that the constant jumps to new and potentially better local minimums can complicate the convergence, as SGD can overshoot and miss the exact minimum.

$$\theta = \theta - \eta \nabla_\theta J(\theta; x^i; y^i) \tag{2.7}$$

Finally mini-batch gradient descent (MBGD), given by equation 2.8, fuses both BGD and SGD and updates the parameter for every mini-batch of n training examples. The resulting advantage is a reduction in the variance of the parameter updates which can lead to a more stable convergence.

$$\theta = \theta - \eta \nabla_\theta J(\theta; x^{i:i+n}; y^{i:i+n}) \tag{2.8}$$

Adaptive learning-rate methods

Adaptive learning-rate methods are optimization algorithms that automatically adapt the learning rate for the presented problem. Some advantages of adaptive learning rates are that it can help against exploding gradients, and can speed up the learning process when the loss function gets stuck on a plateau. Additionally, if the training data is sparse, it can be beneficial to use an adaptive learning-rate method to get the best possible results. Another benefit when using an adaptive learning rate is that the learning rate does not have to be tuned to achieve the best results. Some notable mention of adaptive learning optimization algorithms are Adaptive Moment Estimation known as Adam [58], the Adagrad [27] method and its extension Adadelta [116], and RMSprop proposed by Geoffrey Hinton in a lecture [41].

Back propagation

For ANN with more than two layers, the backpropagation algorithm [90] in conjunction with the gradient descent algorithm is typically used. The backpropagation algorithm is an iterative and recursive method for training ANN.

By using the error backpropagation function, the error is propagated backwards through the network and the error gradient $\nabla E(W)$ is calculated for each layer and determines the adjustments of the weights and biases in the network. The advantages of the backpropagation algorithm are how the weights in the hidden layers between the input and the output of the ANN form itself by recognizing the different patterns in the input data. The backpropagation algorithm is also quite efficient compared to other algorithms. Because of these advantages, backpropagation is the preferred algorithm for supervised learning.

The backpropagation algorithm can be given by four equations [79, pp. c. 2]. The first equation is the equation for the error in the output layer δ^L given by equation 2.9, where L denotes the number of layers in the network, $\frac{\partial C}{\partial a_j^L}$ measure how fast the cost is changing as a function given the j output activation, and $\sigma'(z_j^L)$ measure the rate of change of the activation function σ at the weighted input (z_j^L).

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L) \tag{2.9}$$

The second equation is the equation for the error, δ^l, given the error in the next layer, δ^{l+1}, and is given by equation 2.10 below. Here $(w^{l+1})^T$ is the weight matrix w^{l+1} transposed for the $(l+1)^{th}$ layer, and $\odot\sigma'(z^l$ is the Hadamard product which moves the error backward through the activation function in layer l.

$$\delta^l = ((w^{l+1})^T\delta^{l+1}) \odot \sigma'(z^l) \tag{2.10}$$

The third equation is given by 2.11 is an equation for the rate of change of the cost concerning bias in the network. The fourth and final equation is given by equation 2.12

and is for the rate of change of the cost concerning any weight in the network.

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{2.11}$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^l \delta_j^l \tag{2.12}$$

Given these four equations, we can set up the backpropagation algorithm as follows [79, pp. c. 2]:

1. **Input x:** Set the corresponding activation a^1 for the input layer.
2. **Feedforward:** For each, $l = 2, 3, ..., L$ compute $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$.
3. **Output error δ^L:** Compute the vector $\delta^L = \nabla_a C \odot \sigma'(z^l)$.
4. **Backpropagate the error:** For each $l = L-1, L-2, ..., 2$ compute $\delta^l = ((w^{l+1})^T \delta^{l+1} \odot \sigma'(z^l)$.
5. **Output:** The gradient of the cost function is given by $\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$ and $\frac{\partial C}{\partial b_j^l} = \delta_j^l$.

Vanishing gradient problem

The vanishing gradient problem is a problem that can arise when training ANNs using gradient-based learning methods and backpropagation. As discussed in the previous sections when using gradient descent in conjunction with backpropagation each of the ANNs weights receives an update proportional to the partial derivative of the error function concerning the current weight in each iteration of training, and in some cases, the gradient can get vanishingly small, hence the name. Consequently, this can become a problem as it can prevent the weights affected by changing its value, which can result in halting the learning of the network. The vanishing gradient problem usually occurs in deeper ANNs as the gradient gets smaller when propagating the gradients backwards to the initial layers in the network.

Another similar problem in machine learning is the exploding gradient problem. The exploding gradients problem occurs when significant error gradients accumulate. When this happens, the model may become unstable and impair learning. In the worst-case scenario, the weights become too large and cause a floating-point overflow. When an explosion occurs, the gradient grows exponentially through the network by repeatedly multiplying gradients through the layers having values above 1. The exploding gradient problem is a problem that can sometimes occur in deep networks or recurrent neural network (RNN)s. In these networks, the error gradients accumulate during an update and result in considerable gradients. Consequently, this results in significant updates to the weights of the network and will give an unstable network.

2.5.3 Activation function

Activation functions are functions providing non-linear properties to the neural network, and their main purpose is to convert an input signal of a neuron to an output signal. By using non-linear activation functions, we allow the network to learn more complex and non-linear mappings. An ANN without any non-linear activation functions is simply a linear regression model.

Sigmoid

Figure 2.9 illustrates the sigmoid functions equation and its output. The sigmoid function gives an output in the range from 0 to 1, and is therefore often used in the final layer in ANN for binary classification problems. The output is then determined by a threshold which is usually set to 0.5. If the sigmoid functions output is above the threshold, it classifies as true, and if it is below the threshold, it classifies as false.

A problem with the sigmoid function is the way it compresses an ample input space into the space in the range between 0 and 1. Because of this, even a substantial change in the input of the function will output a small change, which will consequently make the gradient small.

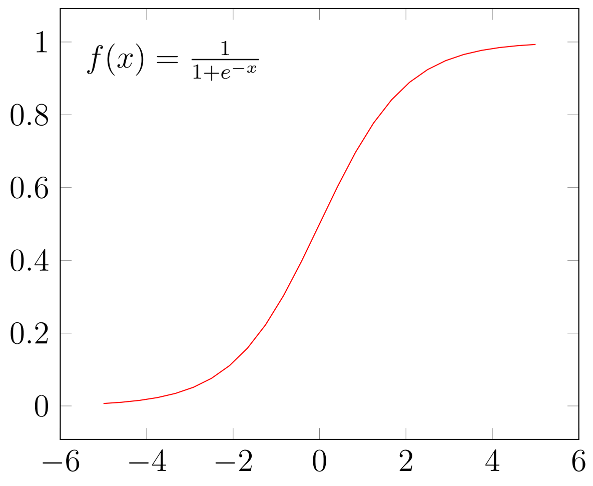


Figure 2.9: Sigmoid

Hyperbolic Tangent

The hyperbolic tangent (TanH) activation function is quite similar to the sigmoid function but gives an output between -1 and 1. The TanH function also suffers the same gradient problems as the sigmoid function because the TanH function compresses the output similarly. The TanH functions equation and output can be seen in the following figure.

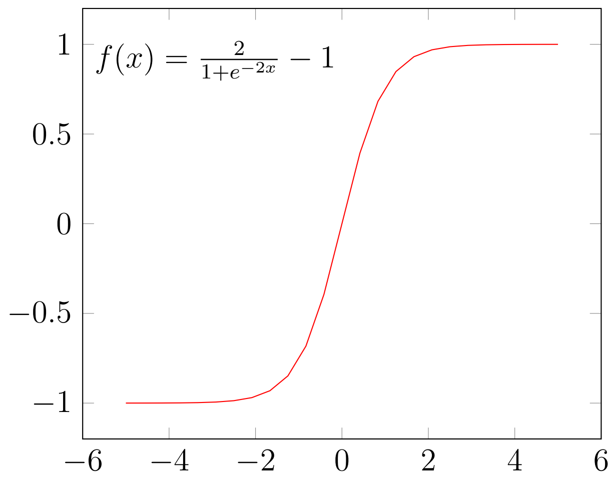


Figure 2.10: TanH

Rectified linear unit

Rectified linear unit (ReLU) activation function gives an output of 0 or real positive value. The ReLU functions output and equation is given in figure 2.11. The advantage of the ReLU function is that it does not suffer as much from vanishing gradient problems. The calculations are also computationally cheap, making the neural network more efficient. The disadvantage of the function is that it removes all negative values making the function unsuitable for some architectures and problems.

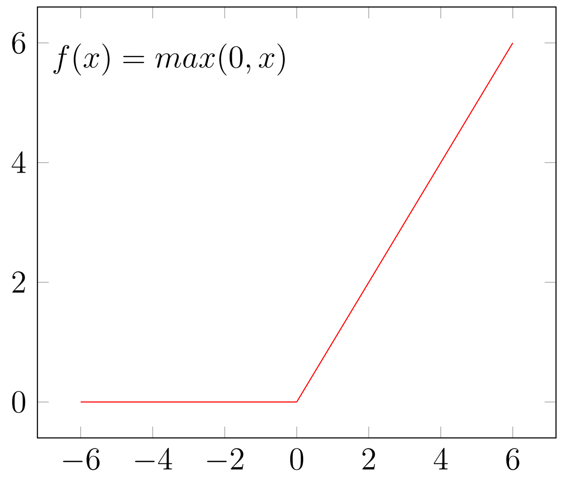


Figure 2.11: ReLU

Softmax

The softmax activation function calculates the probability distribution of each target class over all possible target classes. Further, the softmax function is given by the equation 2.13, where x is the input logits vector. The output range for the softmax function is between 0 and 1, and the sum of all probabilities is equal to 1. Because of this, it is often used in the final layer in an ANN purposed for a classification problem. The output is then chosen by obtaining the index of the vector with the highest probability using the argmax function.

$$f(x) = \frac{e^x}{\sum_i e^{x_i}} \tag{2.13}$$

2.5.4 Model evaluation

The goal of training an ANN is to make it generalize well for never seen before data. There are several methods to measure how well the model generalizes. When evaluating ANNs trained with supervised learning, it is typical to split the dataset into 3 different sets; a training set which is used to train the network, a validation set that is used to verify that the model is generalizing while training, and a test set which is used to confirm the actual performance of the final network. The dataset is typically split to 60-80% to training data, and 20-40% for the testing data.

Overfitting

Overfitting is a problem in ML where the model adjusts itself too closely to the training data [85]. Consequently, this leads to outstanding predictive performance on the training data and data very similar to it, but usually worse otherwise. The opposite of overfitting is underfitting and occurs when a model is not able to adjust itself to the training data. An under-fitted model will usually have poor predictive performance. An overfitted model is usually the result of a model that have too many parameters than necessary. It is usually the opposite for an under fitted model as having too few parameters which make the model too shallow to learn the underlying features of the data. A model can also be overfitted if the training set is sparse as it is not presented with a varied enough dataset, which in turn will make the model worse at generalization.

Early stopping

Early stopping is a technique in machine learning to prevent overfitting. The idea is to stop the training of the network before the weights have fully converged on the training set. Usually, a held-out validation set is used to measure if the performance is getting better or worse, and is used to stop the training if the performance stagnates or decreases.

Cross validation

When dealing with a sparse dataset, it is hard to evaluate how well a model performs. To address this, one can use cross-validation to make a better estimation of how well the model generalizes on unseen data. A commonly used cross-validation method is the k-fold cross-validation method. The original samples are randomly partitioned into k equal-sized sample groups. A single sample group is used as the test data for testing the model, and the remaining $k-1$ sample groups are used as training data. This process is repeated for all of the k sample groups where all the groups are used at exactly once as test data. The results are averaged to produce a single estimation. The final estimation of the model given an evaluation function $f(x)$ is given by the equation 2.14.

$$Average\ Score = \frac{1}{K}\sum_{k=1}^{K} E(W_k) \tag{2.14}$$

2.5.5 Inference

For ML inference is the process of using a trained machine learning model to do the task it was trained for. For example, in this thesis, the task is to segment MRI images. Then the inference process of the models proposed in this thesis will be to segment new MRI images that have not been used in the training process.

2.6 Convolution

Convolution is a mathematical operation that is typically used in digital signal processing (DSP). A more general formula for convolution is given by equation 2.15 where f and g represents the two signals, and t represent time. What we can see from the equation is that a third function is given by the integral of the point-wise multiplication of the two signal functions. This third function is the result of the translation of one of the original functions. Note that this interpretation of the equation represents the operation in the time domain.

$$(f * g)(t) \triangleq \int_0^t f(\tau) * g(t - \tau) d\tau \tag{2.15}$$

For image processing, the formula 2.15 can not be used directly. A discrete non-continuous form of convolution is instead used for digital image processing given by the following equation.

$$(f * g)[n] = \sum_{m=-M}^{M} f[n - m]g[m] \tag{2.16}$$

Where the convolutions are done over two finite sets, $\{-M, -M + 1, ..., M - 1, M\}$. However, this equation gives us an equation for convolution in one dimension. For images the equation 2.17 represents the extended formula for convolution in two dimensions. Also, equation 2.18 shows the extended formula for convolution in three dimensions, which can be used for volumes.

$$(f * g)[m, n] = \sum_{i=-I}^{I} \sum_{j=-J}^{J} f[m - i, n - j]g[i, j] \tag{2.17}$$

$$(f * g)[m, n, o] = \sum_{i=-I}^{I} \sum_{j=-J}^{J} \sum_{k=-K}^{K} f[m - i, n - j, o - k]g[i, j, k] \tag{2.18}$$

The extended equation 2.17 for convolution in two dimensions gives a convolution over two finite matrices given by the following matrix, L.

$$L_{i,j} = \left\{ \begin{matrix} l_{-I,-J} & l_{-I+1,-J} & \cdots & l_{I,-J} \\ l_{-I,-J+1} & l_{-I+1,-J+1} & \cdots & l_{I,-J+1} \\ \vdots & \vdots & \ddots & \vdots \\ l_{-I,J} & l_{-I+1,J} & \cdots & l_{I,J} \end{matrix} \right\} \tag{2.19}$$

For this image, we can see that the convolution operation is done by adding each pixel of the image to its neighbouring pixels and using the kernel matrix to weight them. The operation is basically taking the dot product of a subset of the image with an $n \times n$ matrix. The convolution is applied as a sliding window covering the whole image. The same methods apply for convolution on a volume, but is done by using 3D kernels instead of 2.

Figure 2.12 show this operation done on a 2D matrix using a 3×3 kernel on a single pixel. Figure 2.13 show this convolution applied to all the pixels by sliding the kernel as a window across the image. Notice how the number of border pixels is reduced after the convolution operation. Consequently, this happens because the kernel as a sliding window cannot do convolution on non-existing pixels. A way to avoid this is by adding the borders of the image or volume before the convolution operation. By padding the image before the convolution operation, we can preserve the image dimensions.

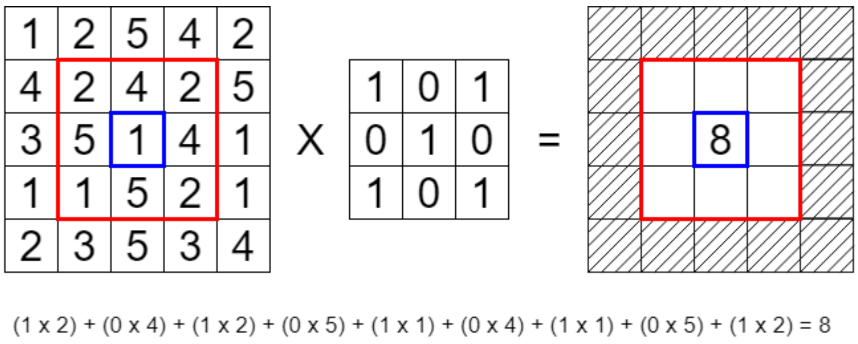

1 2 5 4 2
4 2 4 2 5
3 5 1 4 1
1 1 5 2 1
2 3 5 3 4
X
1 0 1
0 1 0
1 0 1
=
8
(1 x 2) + (0 x 4) + (1 x 2) + (0 x 5) + (1 x 1) + (0 x 4) + (1 x 1) + (0 x 5) + (1 x 2) = 8

Figure 2.12: Convolution kernel operating on a single pixel in an image.

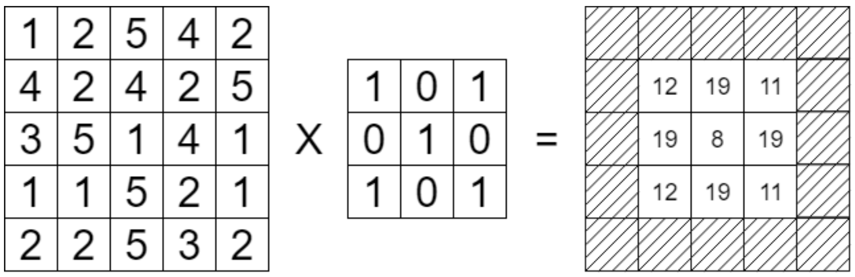


Figure 2.13: Complete kernel operation on all pixels in an image.

2.7 Convolutional neural networks

Convolutional neural network (CNN) is a type of ANN inspired by the visual cortex of the brain, and is commonly used for computer vision problems. The CNN was first proposed by LeCun et al. [113], which were inspired by previous work on the receptive field by Hubel et al. [43]. Further, the CNN have been adapted and been used in a wide range of applications [29, 111, 6].

CNN usually also falls under the category of deep neural network (DNN), as the CNN layer structure is typically very deep. The structure of a CNN is similar to a vanilla feedforward ANN; however, CNN also uses convolutional layers which perform convolution operations on its inputs. In classification tasks, the convolutional layers work as filters extracting the most important features from an image. While for earlier computer vision algorithms the convolution filters had to be engineered by hand, a CNN can learn the parameters for the convolutional filters. Another advantage of CNN is the minimal amount of pre-processing needed for the data. However, this also comes with the cost of being computationally expensive to train.

CNNs are quite similar to standard ANN as they have an input layer, output layer, and hidden layers. However, two properties distinguish CNN from standard feedforward ANN. These properties are weight sharing and local receptive fields and will be further explained in the following subsections.

2.7.1 Weight sharing

In fully connected layers, every connection is weighted by weight, w, and every neuron has a bias, b. Convolutional layers, on the other hand, share the weights and biases as a vector also known as a kernel. These kernels are a filter applied like a sliding window throughout the entire input field functioning like the convolution operations previously discussed in section 2.6. The output of the layer is then a filtered representation of the input, also known as a feature map.

By using the same filters over the whole input field, it applies the same filter to all parts of the input field and detects the same type of features indifferent to the location in the input field. This is an advantage, as the CNN will become invariant to different translations of the features such as location, and illuminance. Another advantage of weight sharing is the reduction of memory usage, as sharing weights significantly reduces the parameters needed to learn the underlying features of the input. This, in turn, makes the CNN less prone to overfitting and makes the training faster.

2.7.2 Local receptive fields

A problem with fully connected ANNs is how the neurons in a layer connect to every neurone in the subsequent layer for every new neuron added to the network the number of connections increases exponentially. As a consequence, the number of parameters will both hinder training and computational throughput. Because of this, using images as input, which usually need to have relatively large dimensions to keep its features, makes it impractical or even impossible to use fully connected layers.

However, for visual problems, it is not necessary to use fully connected layers. By exploiting that pixels in images are typically highly correlated to the adjacent pixels and less correlated to more distant pixels in the image, we can save a substantial amount of computational resources. And is done by connecting the neurons to only a local region of the previous layer. The input area of the neuron is typically called the receptive field of the neuron as it works very similar to the receptive fields in the visual cortex. Compared to a fully connected layer, the convolutional layer uses a receptive field like layout, where each neuron connects to only some of the neurons in the previous layer, instead of the entire previous layer.

In the initial layers, the receptive field of the neurons only encompasses a small area of the image. In contrast, the neurons in the final layers are the combination of multiple receptive fields from the previous layers, have, in a way, extracted a more significant portion of the image. By stacking multiple convolutional layers, the network can learn increasingly abstract features originating from the input image. While the first layer filters typically learn basic features and patterns of the image, like edges, the later layers have more sophisticated filters extracting abstract representations of the image.

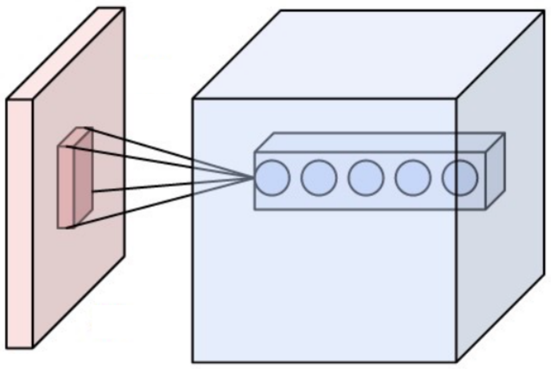


Figure 2.14: Neurons of a convolutional layer (blue), connected to their receptive field (red) [20]

2.7.3 Transposed-convolution

Transposed convolution, sometimes called de-convolution, is an operation typically used in the decoding layer of convolutional autoencoders, or to project feature maps to a higher dimensional space [28]. The transposed convolution works by swapping the forward and backward passes of the convolution operation and making the forward pass generate a higher resolution feature map. Compared to normal up-sampling where the input is just upscaled, the transposed convolution will learn how to upsample the input in the best possible way.

2.7.4 Pooling

In CNNs the pooling layer works as a form of non-linear down-sampling. The pooling layer reduces the spatial size of the input representation, which reduces the number of parameters, which reduces memory usage, and makes the network more computationally efficient. A common theme in CNNs is to insert a pooling layer after a couple of successive convolutional layers to reduce the spatial dimensions of the feature map.

The two most common pooling methods are average pooling and max pooling. A visual example of average pooling is illustrated in figure 2.15, and for max-pooling in figure 2.16. The filter parameters determine the window size of the pooling, and the stride parameters determine how many tiles the window slides over. The stride parameter will also factor the downscale. As an example, a stride of 2 will halve the resolution of the input.

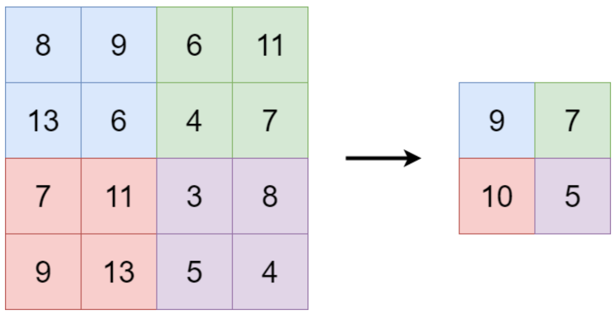


Figure 2.15: Average pooling 2x2 filters and stride 2.

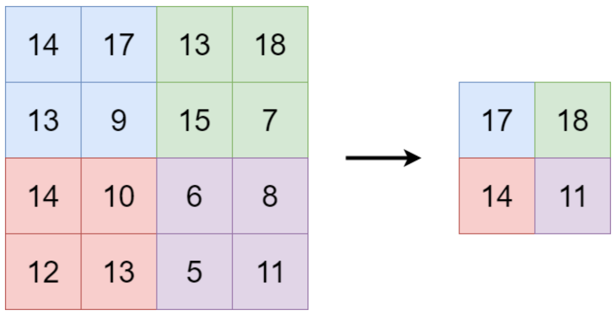


Figure 2.16: Max pooling 2x2 filters and stride 2.

2.8 Autoencoder

An autoencoder is a neural network used to learn efficient data coding and was introduced by Cheng-Yuan Liou et al. [67, 68]. The autoencoder consists of a reducing part (the encoder) and an expanding part (the decoder). The encoder part encodes the input data to a representation of a lower dimension. The decoder tries to generate output from this reduced encoding, resembling the input data as close as possible.

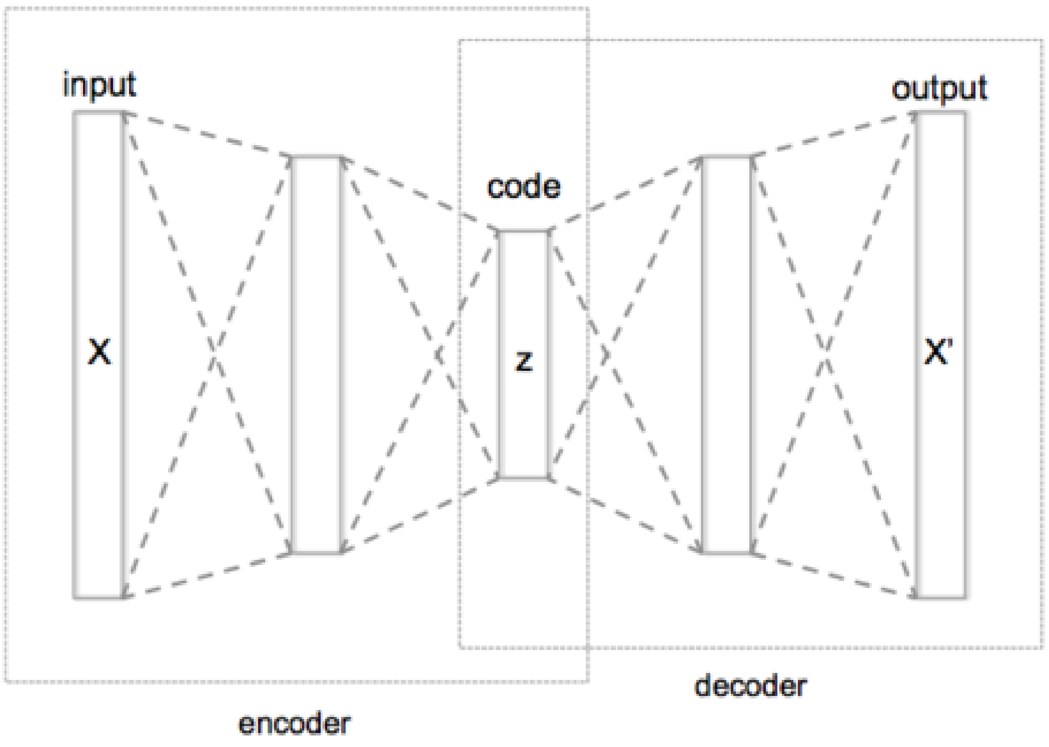


Figure 2.17: Autoencoder structure[3]

2.9 Semantic segmentation evaluation metrics

When evaluating segmentation masks should match the resolution of the ground truth masks, and also match the number of possible classes. When evaluating for several classes, the score is calculated for each class independently and then averaged over all classes to yield a global average score of the prediction mask.

2.9.1 Pixel accuracy

Pixel accuracy is a metric simply evaluating the percentage of pixels in the image or volume that were correctly classified. Equation 2.20 shows the formula for pixel accuracy. Pixel accuracy can be useful for some segmentation tasks. However, it can be a misleading metric when the segment class representation is small within the image or volume because the measure will also be evaluating how well the model identifies cases where the segment class is not present.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.20}$$

2.9.2 Dice Coefficient

The (Sørensen-)Dice coefficient [98, 26], also known as the F1 score, is used to compare the similarity of two samples. The Dice coefficient is commonly used in medical image segmentation [119], as it gives the percentage of overlap between a ground truth mask and a prediction mask. The Dice coefficient is given by the following equation:

$$D(A, B) = \frac{2(A \bigcap B)}{|A| + |B|} \tag{2.21}$$

2.9.3 Jaccard index

The Jaccard index [50, 51], also known as the intersection over union (IoU) metric, is a method which quantifies the present overlap between the ground truth segmentation mask and the prediction mask. The formula for the Jaccard index can be seen in equation 2.22. The number of pixels common between the ground truth and prediction masks divided by the total number of pixels present in both the ground truth and prediction mask.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2.22}$$

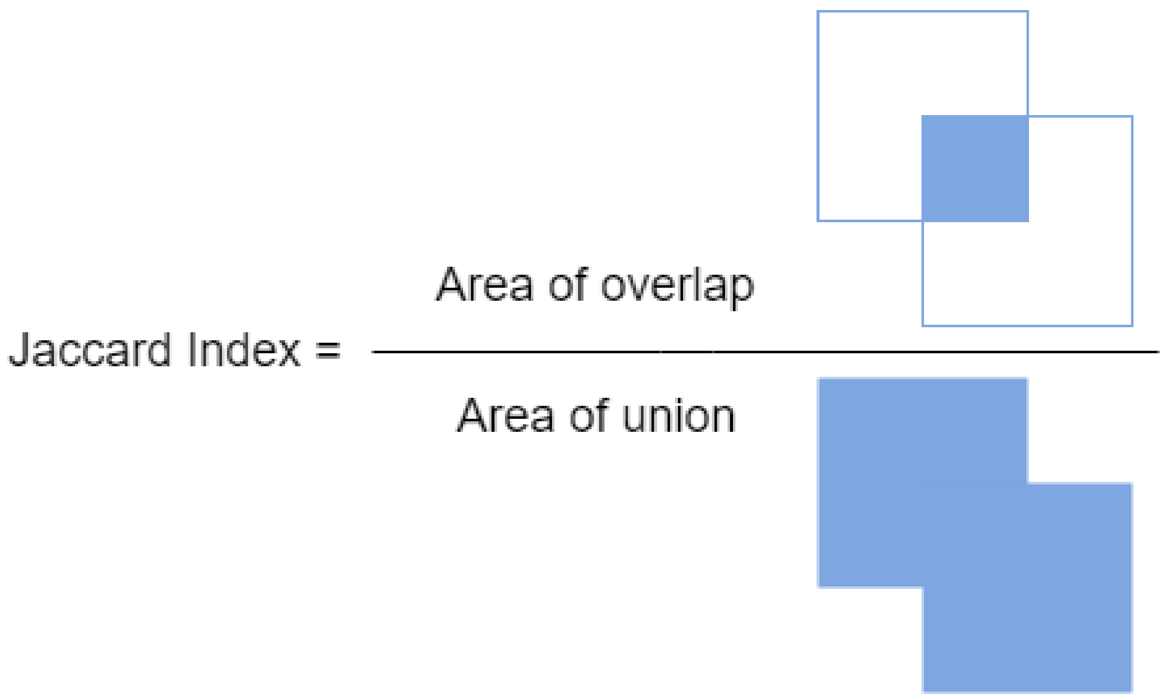


Figure 2.18: Jaccard Index visualized

2.9.4 Machine learning metrics

A table of relevant metrics for both machine learning and segmentation tasks is presented in table 2.1. The metrics are derived from the true positive (TP), false positive (FP), true negative (TN) and false negative (FN) rates.

Description	Derivations
True positive (TP)	TP
True negative (TN)	TN
False positive (FP)	FP
False negative (FN)	FN
Dice coefficient / F1_score	$\frac{2TP}{2TP+FP+FN}$
Jaccard coeffecient / Intersection over Union (IoU)	$\frac{TP}{TP+FP+FN}$
Accuracy (ACC)	$\frac{TP+TN}{TP+TN+FP+FN}$
Sensitivity, recall, hit rate, or true positive rate (TPR)	$\frac{TP}{TP+FN}$
Specificity, selectivity or true negative rate (TNR)	$\frac{TN}{TN+FP}$
Precision or positive predictive value (PPV)	$\frac{TP}{TP+FP}$
Negative predictive value (NPV)	$\frac{TN}{TN+FN}$
Miss rate or false negative rate (FNR)	$\frac{FN}{TN+TP}$
Fall-out or false positive rate (FPR)	$\frac{FP}{TP+TN}$
False discovery rate (FDR)	$\frac{FP}{FP+TP}$
False omission rate (FOR)	$\frac{FN}{FN+TN}$
Informedness or Bookmaker Informedness (BM)	$TPR + TNR - 1$
Markedness (MK)	$BM = TPR + TNR - 1$

Table 2.1: Machine learning metrics

2.10 Data augmentation

Data augmentation is a form of regularization method in ML used to avoid overfitting. The main idea of data augmentation is to increase the size of the training set making ANNs less prone to overfitting. The size of the training set is increased by creating new training samples from the original training samples using augmentation functions. The augmentation functions change the original data to create new data presented in plausible conditions that may occur in the real data. For example, an image classification problem the dataset may have images taken in a limited set of conditions, but the target of the classification could exist in a larger variety of conditions. These conditions could, for example, be various locations, orientations, scale, contrast, brightness. By training an ANN on data with these new conditions, it becomes invariant to translations, which makes the ANN more robust and better able to generalize when presented with new data.

However, when augmenting data, it is essential to know that the ANN could be sensitive to some types of augmentations for a given problem. For example, flipping the image when the model should be able to classify between a six and a nine in an image classification problem would not be a good idea because of this, it is essential to know which augmentations methods to use, and how to set the right augmentation parameters to generate samples set in plausible conditions.

2.11 Ensemble

The goal of machine learning is to find an optimal model that is best suited for a given problem. Instead of training one single model, we can combine multiple ML models, which are known as ensemble learning. For ML ensemble methods use multiple learning algorithms to get superior prediction results than those from using a single learning algorithm. Some of these ensembles methods are given below:

- **Bootstrap aggregation (BAGGING)** [8] is an ensemble method where a set of models is trained on the training data. Then use the models vote with equal rights. The votes are cast by aggregating the outputs of the models. To the best possible variance for the models, the models should be trained using various subsets of the data, different models, and different parameters.
- **Ensemble averaging** is an ensemble method where multiple models are created and combined to produce an average output. The intuition is that the ensemble will average out the errors of the models, and will produce a superior output than any of the individual models.
- **Bucket of models (BoM)** is an ensemble method where an algorithm used to choose the best model from the "bucket" for each problem. The main idea of this method is that when presenting many different problems, it will typically give a superior result generally than using a single model from the "bucket."
- **Stacking** [109] is an ensemble method combining the predictions of several learning algorithms. The method is using a combiner learning algorithm trained to make the final prediction using predictions from the previously trained algorithm on the training data as additional inputs to the training data.

2.12 Parameter search

A parameter search is a search for the best parameters for a model by minimizing a predefined loss function given some input data [17]. In machine learning, a parameter search is often referred to as hyperparameter optimization, where a hyperparameter is a parameter used to control the learning process.

Parameter searches can be performed using simple methods like grid search, where a grid of parameters is tested incrementally, or random search where the value of the parameter is randomly assigned within their set range or parameter alternatives. More advanced parameter search methods include optimization algorithms like genetic algorithm (GA) [77] and particle swarm optimization (PSO) [35, 114], where these methods use more sophisticated methods to find the best parameters quickly.

2.13 Conditional random field

In 2001 John Lafferty et. al [62] presented a method for segmenting and labelling sequence data called conditional random field (CRF). CRFs are types of discriminative undirected probabilistic graphical models. For image segmentation the CRF potentials incorporate smoothness terms that maximize label agreement between similar pixels, and can integrate more elaborate terms that model contextual relationships between object classes. John Lafferty et al. [62] defines a conditional random field as follows:

Definition. *Let $G = (V, E)$ be a graph such that $\mathbf{Y} = (\mathbf{Y}_v)_{v \in V}$, so that $\mathbf{Y}$ is indexed by the vertices of G. Then $(\mathbf{X}, \mathbf{Y})$ is a conditional random field in case, when conditioned on X, the random variables $\mathbf{Y}_v$ obey the Markov property with respect to the graph: $p(\mathbf{Y}_v|\mathbf{X}, \mathbf{Y}_w, w \neq v) = p(\mathbf{Y}_v|\mathbf{X}, \mathbf{Y}_w, w \sim v)$, where $w \sim v$ means that w and v are neighbors in G.*

By this definition, we see that a CRF is an undirected graphical model where its nodes are divided into two disjoint sets, the observed variables $\mathbf{X}$, and the output variables $\mathbf{Y}$. This gives the modeled conditional distribution $p(\mathbf{X}|\mathbf{Y})$.

2.14 Marching cubes algorithm

The marching cubes algorithm is an algorithm used for mesh generation from three-dimensional isosurfaces and was published in 1987 by Lorensen et al. [70]. The algorithm is quite often used in medical imaging to visualize segmented MRI images in 3D. How the algorithm works can be summarized as follows [70]:

1. Read four slices into memory.
2. Scan two slices and create a cube from four neighbours on one slice and four neighbours on the next slice.
3. Calculate an index for the cube by comparing the eight density values at the cube vertices with the surface constant.
4. Using the index, look up the list of edges from the pre-calculated table.
5. Using the densities at each edge vertex, find the surface edge intersection via linear interpolation.
6. Calculate a unit normal at each cube vertex using central differences. Interpolate the normal to each triangle vertex.
7. Output the triangle vertices and vertex normals.

Chapter 3: Related work

This chapter will discuss some of the work related to this project. Much work and effort have been devoted to research regarding the segmentation of biomedical images. Moreover, in recent years, a substantial amount of development has happened in the fields of ML and CNN. Section 3.1 will discuss various convolutional neural networks for object classifications, which have laid the foundation for convolutional neural networks regarding semantic segmentation, which will be further discussed in section 3.2. In section 3.3 loss functions dealing with high segment class imbalance will be discussed. And finally, a method building on CRF models will be discussed in section 3.4

3.1 Convolutional neural network architectures for object classification

In recent years, several architectures for object classification using CNNs have emerged. CNNs have proven to be excellent tools for computer vision tasks. In this section, we will examine some of the most reputable CNN architectures. The networks presented have also inspired many of the networks for segmentation, which will be further discussed in section 3.2. Many of the CNN architectures discussed in this section developed around and submitted to the ImageNet Large Scale Visual Recognition Challenge 2014 (ILSVRC14) [91]. ILSVRC14 [91] is a multitask challenge where one of the tasks is to get the best classification accuracy on a supplied dataset containing over ten million images consisting of more than 1000 classes. This competition has been a huge factor in the development of CNNs recently.

3.1.1 LeNet5

LeNet5[113] is one of the CNN architectures that sparked the interest in CNNs. The network was developed by Yann LeCun in 1998 and was used for digit recognition. Figure 3.1 shows the LeNet5s architecture. LeNet comprises seven layers and alternates between convolutional layers and pooling layers, except for the input layer, output layer, and the layer before the output layer. The network is quite shallow compared to newer CNNs, which reflects the computer hardware at the time.

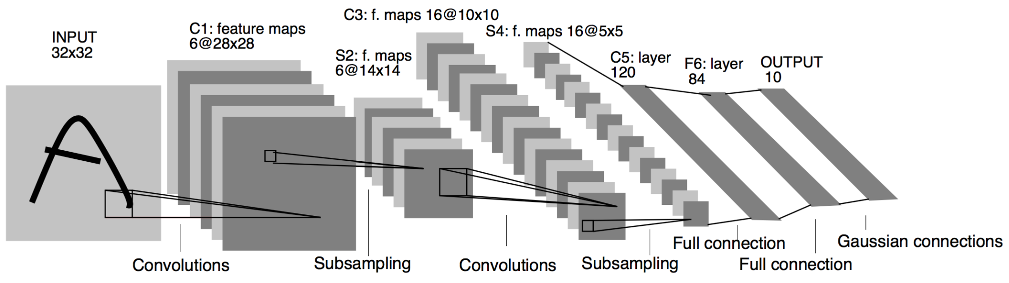


Figure 3.1: LeNet5 architecture[113]

3.1.2 AlexNet

AlexNet [61] is considered one of the most important contributions to modern machine learning. The network introduced many new techniques to the field, such as the use of the ReLU as an activation function, the method of stacking multiple convolutional layers before the pooling layers, the use of max pooling and its benefits over the use of average pooling, and the introduction of graphical processing unit (GPU) training.

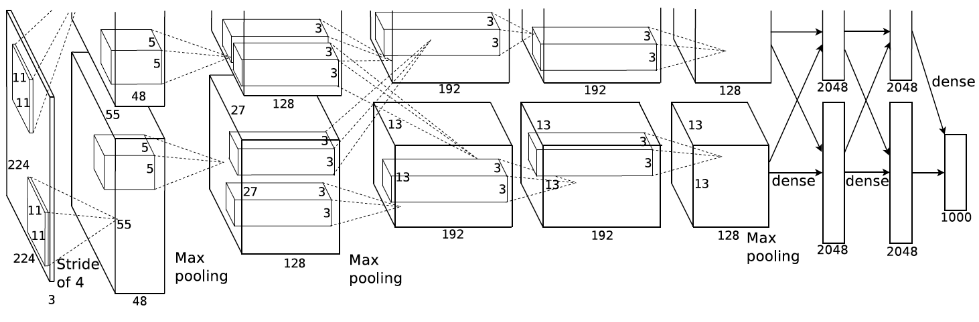


Figure 3.2: AlexNet architecture[61]

3.1.3 VGGNet

In 2014 the VGG Network (VGGNet) [96] was developed by the VGG from the University of Oxford. The network came in 2nd place of the ILSVRC14 [91] classification task and 1st in the localization task. The architecture of VGGNet is quite similar to AlexNet[61], and is the most preferred choices in the ML for image feature extraction. VGGNet uses only 3×3 convolution and 2×2 pooling throughout the whole network. The reason for this is that by using 2 layers of 3×3 filters, it has already covered a 5×5 area, and by using 3 layers of 3×3 it covers an effective area of 7×7. Thus, large filter sizes, such as 11×11 in AlexNet [61], are unnecessary. The use of smaller filter sizes also leads to reduction in number of parameters which gives faster convergence and reduced likelihood of overfitting.

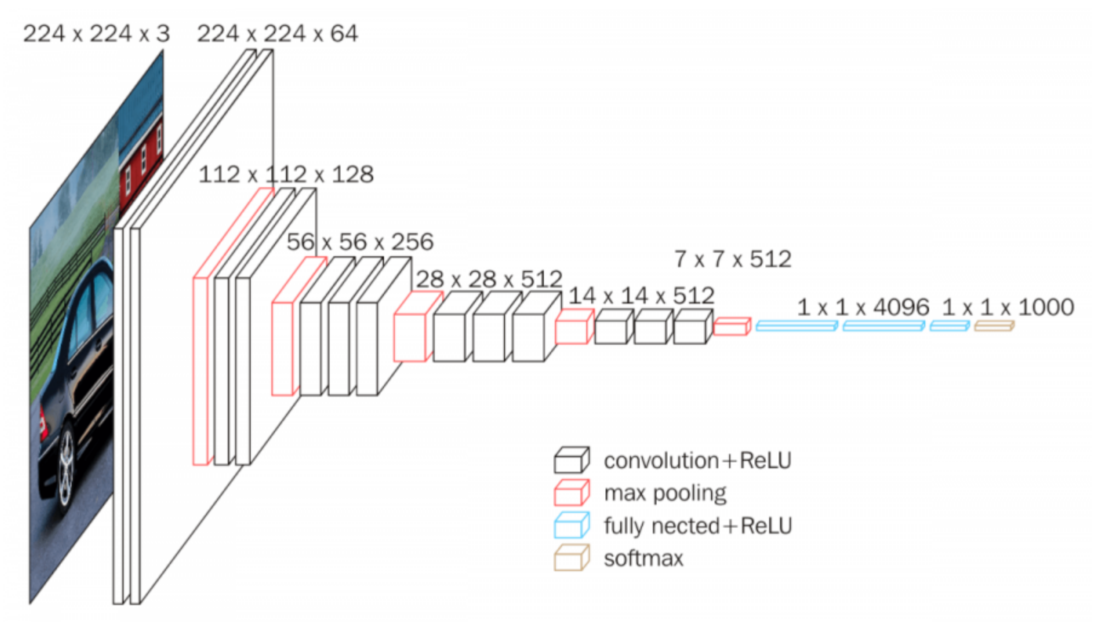
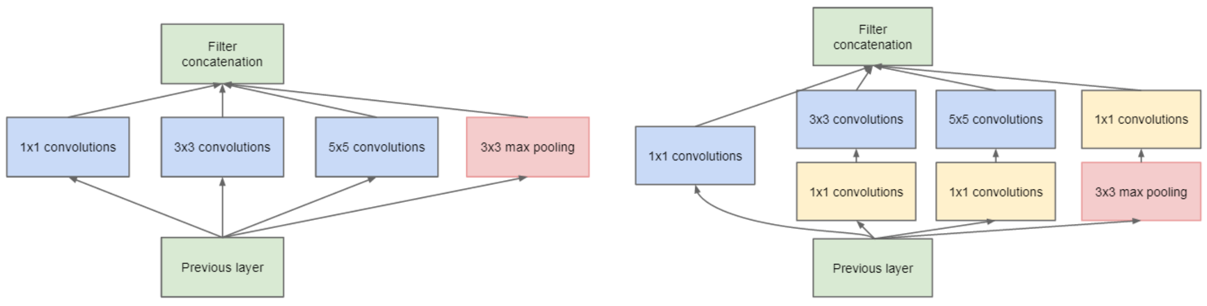


Figure 3.3: VGG 16 architecture[96]

3.1.4 Inception

In 2014 Szegedy et al. released the GoogLeNet [102], which won the ILSVRC14 [91]. The name is a play on words paying tribute to the LeNet [113] by Yan LeCun. Inspired by the VGGNet [96] methods for reducing the number of parameters, the GoogLenet introduces some new methods. To further reduce the number of parameters, a global average pooling operation layer was used instead of fully connected layers in the final layers. Another method introduced, is the use of 1×1 convolution reduce the computation bottleneck. The 1×1 convolution is used before more expensive blocks and works as a dimension reduction module which reduces the computation needed. This method made the Inception module possible and is illustrated in figure 3.4.



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 3.4: Inception modules [102]

3.1.5 Residual neural networks

In 2015 He et al. [39] at Microsoft Research released the residual neural network (ResNet), which won the ImageNet Large Scale Visual Recognition Challenge 2015 (ILSVRC15). Notably, the ResNet introduced the skip connections layers, which can be seen in figure 3.5. The skip layers make the layers reuse activations from a previous layer until the adjacent layer learns its weights, which ultimately simplifies the network by using fewer layers in the initial training stages. Consequently, this will speed up the training because of the reduction of vanishing gradients. The skip connections made it possible to train CNNs with over 1000 layers.

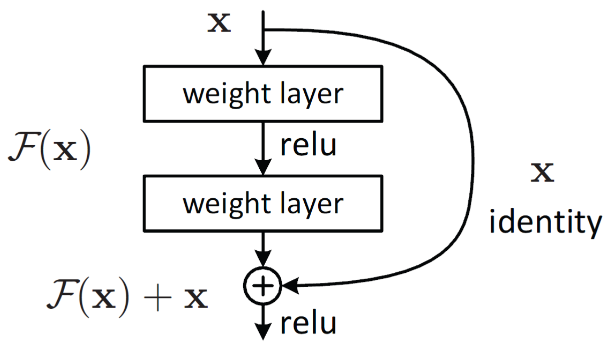

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 3.5: Residual block [39]

Aggregated Transformations

In 2016 Xie et al. [110] released a follow up on the ResNet called the ResNext network. The network proposes a block of operations, where the block is split into multiple paths. ResNext also introduces a new dimension called cardinality and is the number of paths in the block. The block takes inspiration from the inception block mentioned in section 3.1.4, and uses a similar split-transform-merge paradigm. The difference is that the ResNext block merges the output paths by adding them together instead of concatenating them. Another difference is that the path uses the same topology. The block also makes use of the skip connections from the ResNet architecture. The proposed ResNext block can be seen in figure 3.6.

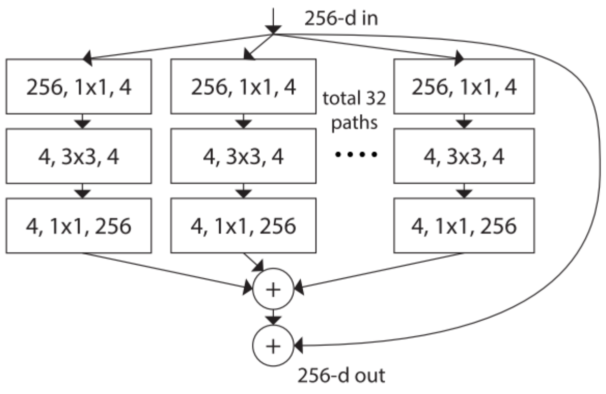


Figure 3.6: A Block of ResNeXt with Cardinality = 32 [110]

3.2 Convolutional neural network architectures for semantic segmentation

This section will focus on CNN architectures for pixel and voxel-wise segmentation of images and volumetric data. As we will see, these networks are inspired by and build upon the networks for object classification discussed in the previous section.

3.2.1 Fully convolutional networks for semantic segmentation

Long et al. developed a FCN for semantic segmentation in 2015 [69]. Their approach draws on previous CNNs for classification tasks, specifically the AlexNet[61], VGGNet [96], and GoogLeNet[102]. The main difference to a regular CNN is that the fully connected layer at the very end (which are typically used for classification of the image features) is replaced with convolutional layers, enabling the network to output a heatmap (segmentation map) instead of classification as seen in figure 3.7. Another feature of the FCN are the skip-connections added between layers to fuse coarse, semantic and local appearance information. The skip connection architecture is learned from end-to-end and refines the semantics and spatial precision of the output.

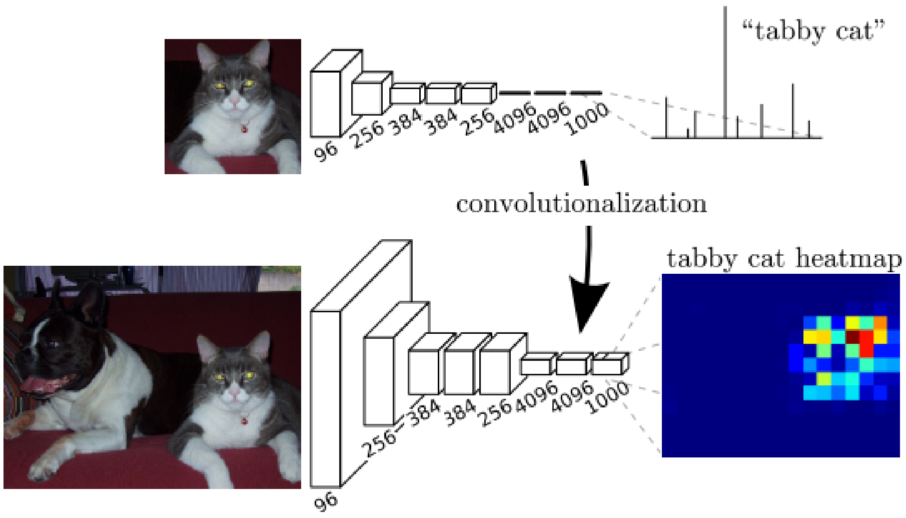


Figure 3.7: Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. [69]

3.2.2 V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation

In 2016 Milletari et al. [75] proposed an approach to 3D volume segmentation based on a volumetric FCN. The proposed network is called V-Net, and was developed to segment the prostate in MRI images taken around the prostate area with a fixed size of $128 \times 128 \times 64$ voxels and a spatial resolution of $1 \times 1 \times 1.5$ millimetres.

Similar to the structure of an autoencoder, the V-Net network has a contracting path and an expanding path. The contracting part of the network is divided into three stages. All the stages consist of three convolutional layers. The convolutional layers in each stage use 3D kernels with the size $(5 \times 5 \times 5)$ voxels, and the number of feature channels doubles at each stage. For each stage, the input of the block is also added to the output of the last convolutional layer to enable learning a residual function. The data resolution through the contracting part is reduced for each stage, but instead of using pooling to down-sample the data, a convolutional layer with kernel $(2 \times 2 \times 2)$ voxels with a stride of two is used. It is argued that the memory footprint is reduced during training because there is no need for switches mapping the output of pooling layers back to their inputs for the back-propagation.

Similar to the U-Net architecture, which will be further discussed in section 3.2.3, the network also has skip-connections going from the contracting part of the network to the expanding path for each stage. The skip paths are used to gather the fine-grained detail that otherwise would be lost in the contracting part of the network, and is for improving the quality of the final segmentation map. The skip paths also improve the convergence time when training the model.

The expanding part of the network is quite similar to the contracting part of the network. The expanding part is almost the inverse of the contracting part. Similar to the down-sampling layers in the contracting part, a transposed convolution operation is used instead to up-sample the resolution of the data after each stage of the expanding part. Also, following the inverse pattern of the contracting part of the network, the number of feature maps is halved after each up-sample. The skip paths are also concatenated after each up-sample with its corresponding down sample layer. The output layers have a $(1 \times 1 \times 1)$ kernel size and are converted to a segmentation map of the foreground and background regions by applying voxel-vice soft-max with the same volume resolution as the input size. The V-Net network architecture is illustrated in the following figure.

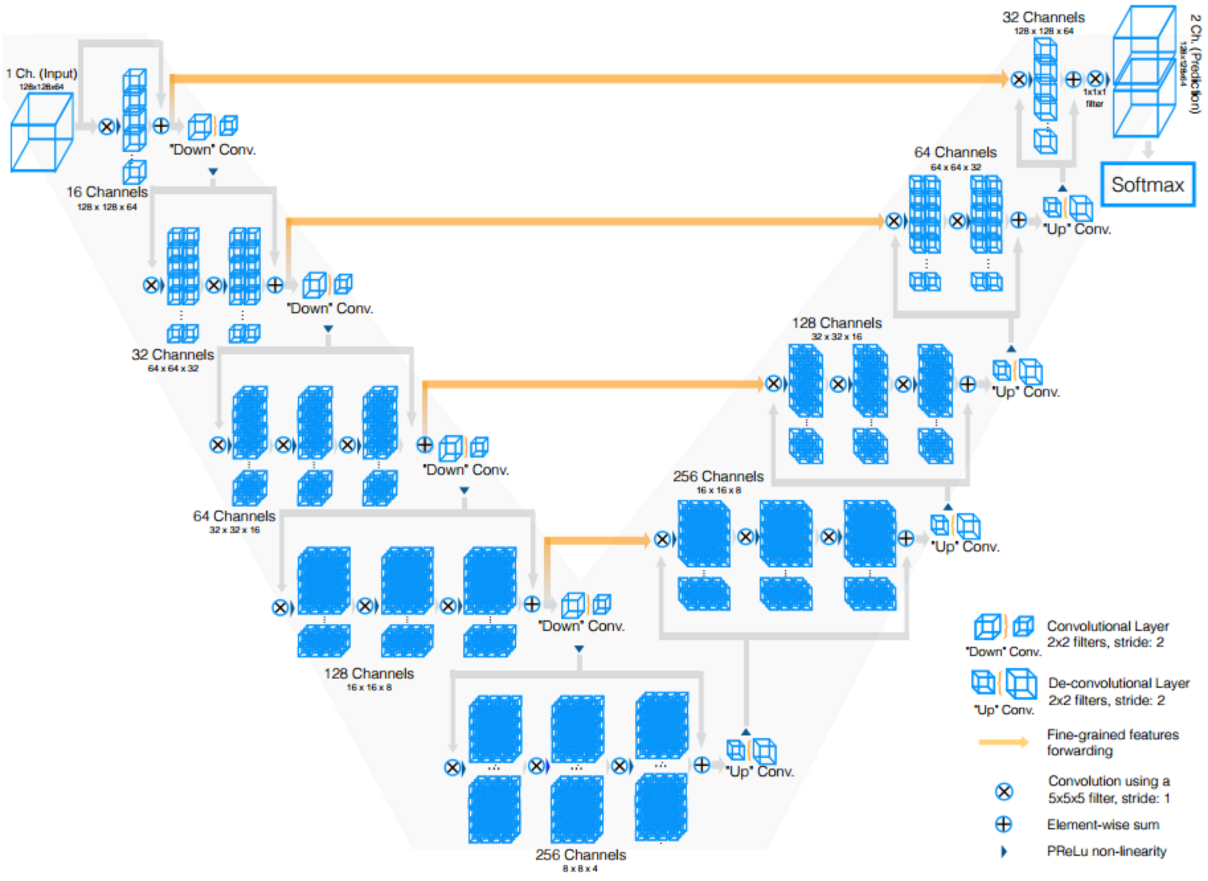


Figure 3.8: V-Net architecture [75]

Dice loss

It is not uncommon that the anatomy to be segmented occupies a tiny region of the volume or image. As a consequence, the learning process frequently gets stuck in a local minimum, which gives a network biased towards the background segment class. Because of this problem, the authors of the V-Net [75] architecture is proposing a loss function for dealing with the imbalance between background and foreground voxels. The loss function is based on the Dice coefficient, discussed in section 2.9.2. The equation for the dice loss function for two binary volumes is given by the following equation:

$$D = 1 - \frac{2\sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2} \tag{3.1}$$

where p is the predicted mask, g the ground truth mask for the ith class of N classes, the authors further argue, weighting the different segment classes is unnecessary when using this loss function.

3.2.3 U-Net: Convolutional Networks for Biomedical Image Segmentation

In 2016 Ronneberger et al. [87] proposed a CNN architecture for segmenting biomedical images. The U-Net architecture builds on the FCN architecture [69] as previously discussed. The U-Net architecture is illustrated in figure 3.9. The network has a similar structure to an autoencoder, and the previously discussed V-Net [75], as it consists of a contracting and an expanding part connected by skip connections.

The contracting part of the network consists of a repeated pattern of stages involving convolutions and pooling operations. The stages use two consecutive 3×3 convolutional layers before the feature channels are doubled by a max-pooling layer. The expanding part has a similar pattern to the contracting part except the pooling layers are replaced by transposed convolutional layers, and the feature channels are halved for each up-sampling stage. The expanding path also features concatenation layers that connect the correspondingly cropped feature map from the contracting path. These concatenations establish skip connections that allow feature representations to pass through the bottleneck. The skip connections combine the localized features from the contracting path with the more contextual features gathered in the expanding path. This combination helps the network generalize better.

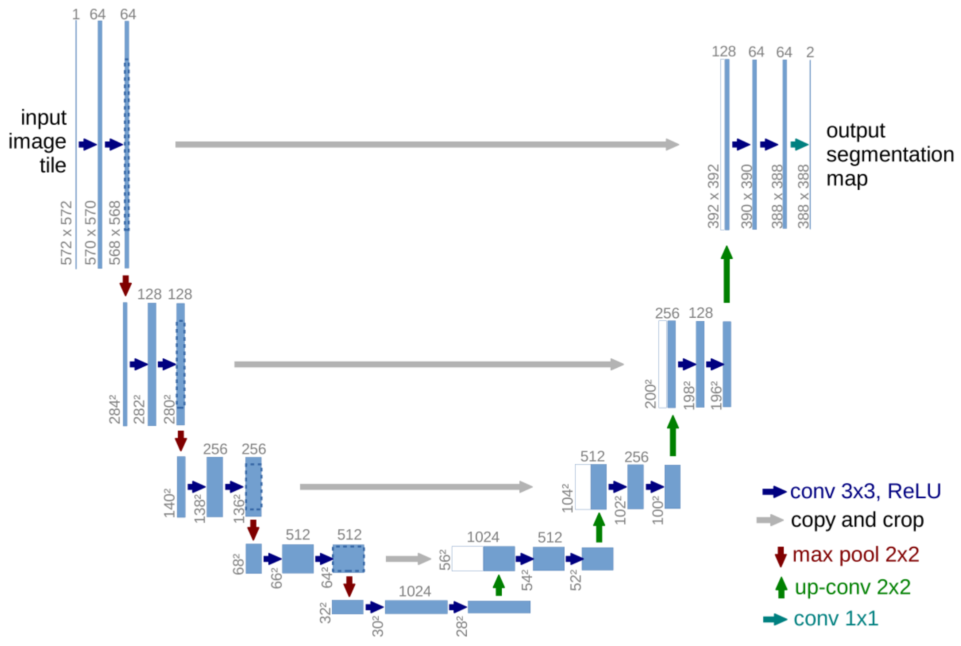


Figure 3.9: U-Net architecture [87]

Weight map loss

The authors of the U-Net architecture also introduce a weight-map loss function, which gives some pixels higher importance in training. A weight map is pre-calculated for each ground truth segmentation to compensate for the different class frequencies in the training set. Separation borders are also calculated using morphological operations, as it is useful when segmenting touching objects of the same class. Function 3.2 show how the weight map is computed.

$$w(x) = w_c + w_0 \cdot exp(-\frac{(d_1(x) + d_2(x))^2}{2\sigma^2}) \tag{3.2}$$

Where $w(x) : \omega \to \mathbb{R}$ is the weight map, $w_c : \omega \to \mathbb{R}$ is the weight map to balance class frequency, $d_1 : \omega \to \mathbb{R}$ denotes the distance to the border of the nearest cell and $d_2 : \omega \to \mathbb{R}$ the distance to the border of the second nearest cell. Figure 3.10 shows an example of how a training sample is weighted.

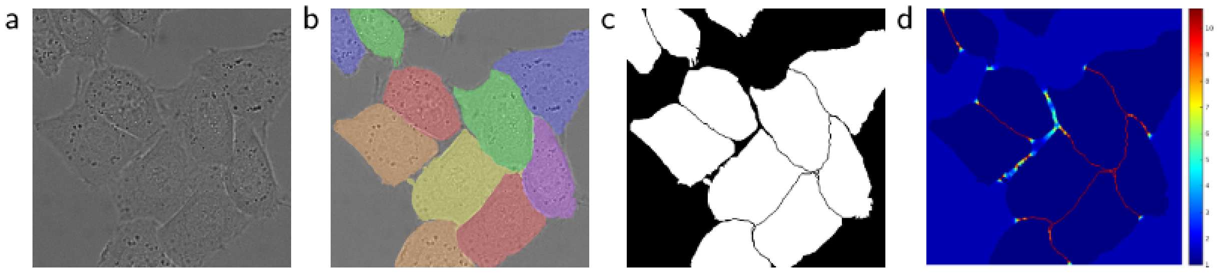


Figure 3.10: HeLa cells on glass recorded with DIC (differential interference contrast) mi- croscopy. (a) raw image. (b) overlay with ground truth segmentation. Different colors indicate different instances of the HeLa cells. (c) generated segmentation mask (white: foreground, black: background). (d) map with a pixel-wise loss weight to force the network to learn the border pixels. [87]

Elastic deformation

The authors also proposed a method for augmenting images or volumes. The method described works as follows: First, a coarse displacement grid with a random displacement for each grid point is generated. Then the grid is interpolated to compute a displacement for each pixel or voxel in the input data. Finally, the input data is deformed using the displacement vectors and spline interpolation.

Because the elastic deformation method outputs plausible images or volumes by applying rigid transformations and slight elastic deformation, the U-Net architecture can learn to generalize very well from only a few annotated samples.

3.2.4 U-Net Variants

Because of the U-Net's [87] versatility and performance of segmenting biomedical images, there have been several noteworthy iterations and implementations of the U-Net which improve upon the original U-Net architecture.

3D U-Net

The 3D U-Net by Cicek et al. [16] is a network based on the U-Net [87] network, but instead for taking 2D images as input it takes 3D volumes. The 2D convolution, 2D up-convolution, and 2D pooling layers are replaced by 3D convolutions, 3D transposed convolutions, and 3D pooling layers. The network follows the same contracting and expanding pattern as the original U-Net, but the number of stages is reduced from 5 to 4. The network also improves on the U-Net architecture by avoiding the bottleneck as suggested in [103]. The bottleneck is avoided by doubling the number of kernels before max pooling in the contracting path. The same scheme is applied in the expanding path.

Further, the authors used a weighted cross-entropy loss function where the weight of unlabelled pixels are set to zero, which they claim makes the network learn from the labelled voxels, and generalize better to the whole volume. Finally, their proposed architecture scores an average IoU of 0.863 in a 3-fold cross-validation experiment with a semi-automated segmentation of a 3D volume containing structures of the Xenopus kidney. A 2D U-Net which segment the volume slice by slice is used for comparison and got an average IoU of 0.796.

Attention U-Net

Oktray et al. [82] proposed a U-Net variant which introduces attention gate that builds on the approach proposed by [52]. The attention gates seen in figure 3.11 are incorporated into the original U-Nets architecture to highlight salient features passed through the skip connections. The gates act as filters, filtering out irrelevant and noisy responses in the skip connections. The gates are connected right before the concatenation operation to merge only the relevant activations.

The proposed architecture is tested against the standard U-Net architecture on a multi-class abdominal CT segmentation dataset, where the goal is to segment the pancreas. It is observed that the Dice score can be increased by 2-3% by using attention gates in the U-Net.

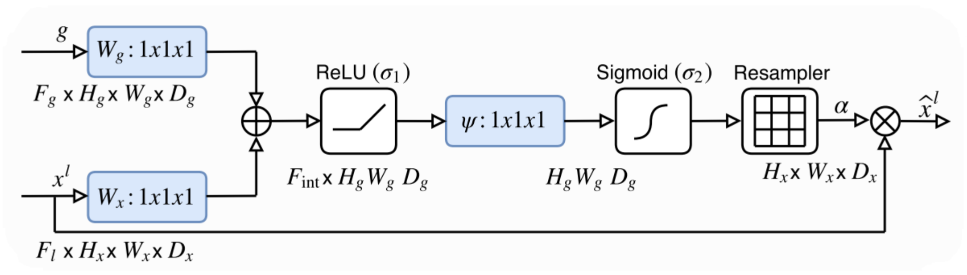
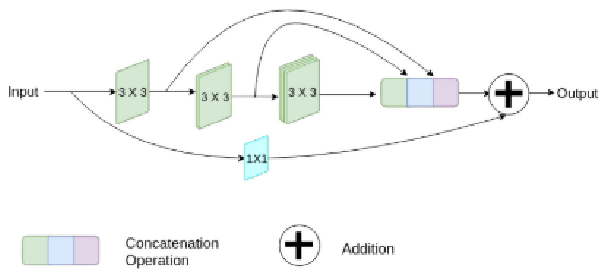


Figure 3.11: Attention gate [82]

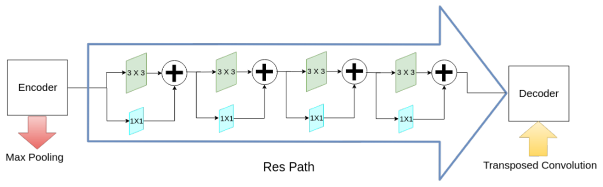
Multires U-Net

Ibtehaz et al. [46] proposes a variant of the U-Net architecture which replaces the standard convolutional layers of the U-Net with a customized block they call *'Multires block'*. The block is inspired by the Inception block [102], but instead of using different sized filters, they only use a succession of 3x3 filters. A skip layer connection [39] is also made between the input and output of the block. The *'Multires block'* can be seen in figure 3.12a.

To alleviate the disparity between the encoder-decoder features, they propose to incorporate convolutional layers along with the shortcut connection of the U-Net. Instead of simply concatenating the feature maps from the encoder stage to the decode stage, the feature maps are passed through a chain of convolutional layers with Resnet-like skip layers before concatenation with the decoder features. This path they call the *Res path*, and is illustrated in figure 3.12b.The proposed architecture was tested on five datasets and showed an improvement of 10.15%, 5.07%, 2.63%, 1.41%, and 0.62% over the standard U-Net.



(a) MultiRes Block



(b) MultiRes Path

Figure 3.12: MultiRes U-Net building blocks [46]

Nested U-Net

In 2018 a Zhou et al. [118] proposed the U-Net architecture. Noteworthy to the architecture is that the U-Net skip pathways are re-designed. The re-designed skip pathways transform the connectivity of the encoder and decoder parts of the networks. The main idea motivation for the re-design of the skip pathways is to bridge the semantic gap between the feature maps of the encoder and decoder before fusion. The Unet++ architecture is illustrated in figure 3.13. The Unet++ architecture was evaluated on four datasets, including lung nodule segmentation, colon polyp segmentation, cell nuclei segmentation, and liver segmentation. The network was compared to the original U-Net and a wider U-Net and increased the average IoU score of 3.9 and 3.4 points.

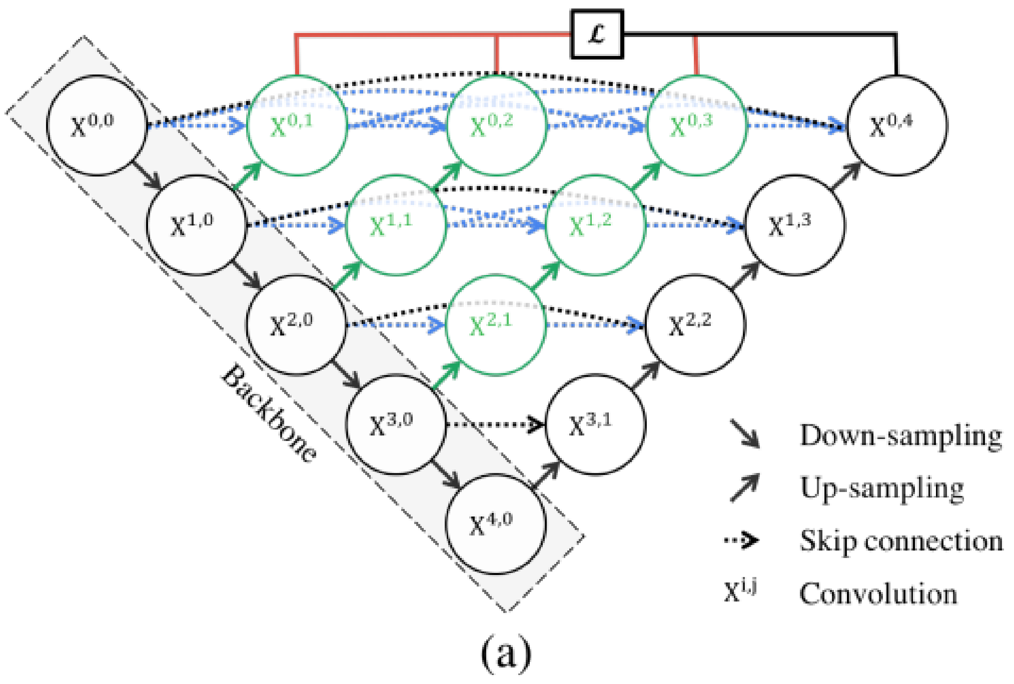


Figure 3.13: U-Net++ architecture [118]

3.2.5 Deep convolutional neural network for segmentation of knee joint anatomy

In 2018, Zhou et. al[117] proposed a quite extensive method using FCN and DCRF to automatically segment knee MRI images. The network they propose follows the autoencoder architecture with a contracting and an expanding path, and layers based on the VGG architecture [96]. Their network, named CED, is trained to predict image data slice by slice from a 3D volume. For inference, a 3D probability map is generated from the stack of 2D probability maps. Then, a fully connected 3D conditional random field, also known as a DCRF [60], is used to improve the 3D probability map. The final segmentation masks are then used to generate a 3D model from the segmentation map. The CED network architecture can be seen in figure 3.14, and the full segmentation pipeline can be seen in figure. 3.15.

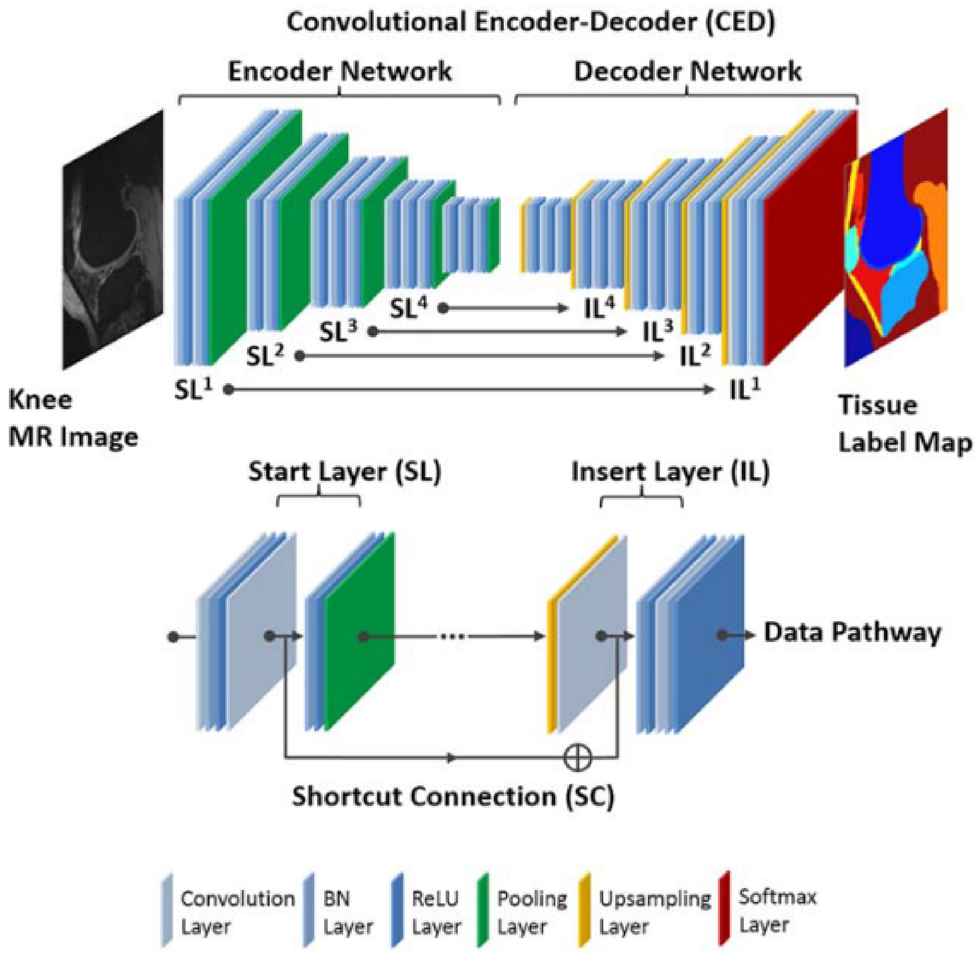


Figure 3.14: CED architecture [117]

The data consisted of fast spin echo (FSE) MRI sample volumes of knee joints from 20 subjects. The samples were segmented into multi-class masks with 13 unique segment classes, including background, femur, femoral cartilage, tibia, tibial cartilage, patella, patellar cartilage, meniscus, quadriceps, and patellar tendons, muscle, synovial fluid-filled joint effusion and Baker's cyst, infrapatellar fat pad, and other non-specified tissues. For training, the dataset was evaluated using k-fold cross-validation and was split into 20 folds, leaving only one sample for evaluation. All musculoskeletal tissues after the full process had a mean Dice coefficient above 0.7.

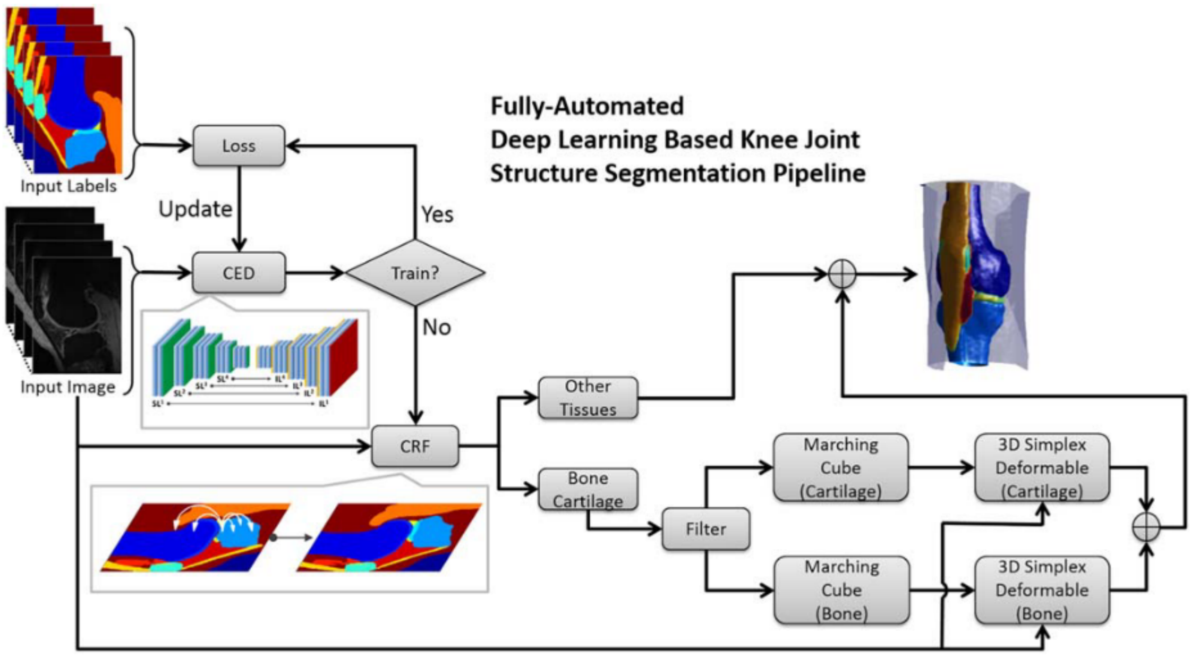

Fully-Automated
Deep Learning Based Knee Joint
Structure Segmentation Pipeline
Input Labels
Loss
Update
Yes
CED
Train?
No
Input Image
Other Tissues
CRF
Bone Cartilage
Filter
Marching Cube (Cartilage)
Marching Cube (Bone)
3D Simplex Deformable (Cartilage)
3D Simplex Deformable (Bone)

Figure 3.15: CED pipeline [117]

3.3 Loss functions for imbalance in segmentation

A problem with medical imaging is how the sizes of the anatomical features can vary. Some of the features are very small compared to the rest of the input volume. This problem translates directly into segmentation, as the segmentation masks for the smaller classes become equally small. Because of this imbalance, it can be hard to train the networks to recognize these features, as they vanish when presented in the bigger picture. In this section, we will discuss various loss functions and methods for trying to handle the issue of class imbalance in segmentation.

3.3.1 Generalized dice loss

Sudre et al. [100] released a paper in 2017 concerning a loss function for dealing with highly unbalanced segmentations. The loss function is a generalized Dice loss proposed by Crum et al. [23]. The function is given by equation 3.3, where r is the ground truth segmentation mask, p_{ln} is the predicted probability map, and w is used to weight the various segment classes calculated using $w_l = 1/(\sum_{n=1}^{N})^2$.

$$GDL = 1 - 2\frac{\sum_{l=1}^{2} w_l \sum_n r_{ln} p_{ln}}{\sum_{l=1}^{2} w_l \sum_n r_{ln} + p_{ln}} \tag{3.3}$$

The generalized dice loss (GDL) function is compared against 3 other loss functions, a 2 class dice loss $\mathbf{DL}_2$ based on the dice loss function described by Milletari et al. [75] mentioned in section 3.2.2, the Sensitivity - Specifity (SS) function described by Brosch et al. [10] given by equation 3.4, where λ weights the balance between sensitivity and specificity and ϵ is a smoothing factor dealing with cases of division by 0, and a weighted cross-entropy (WCE) function given by the equation 3.5.

$$SS = \lambda\frac{\sum_{n=1}^{N}(r_n - p_n)^2 r_n}{\sum_{n=1}^{N} r_n + \epsilon} + (1-\lambda)\frac{\sum_{n=1}^{N}(r_n - p_n)^2(1-r_n)}{\sum_{n=1}^{N}(1-r_n) + \epsilon} \tag{3.4}$$

$$WCE = -\frac{1}{N}\sum_{n=1}^{N} w r_n log(p_n) + (1-r_n)log(1-p_n) \tag{3.5}$$

The loss functions are tested on 4 different patch-based networks, the 2D networks U-Net [87] and TwoPathCNN [38], the 3D networks DeepMedic [54] and HighResNet [66]. The 2D networks are tested on the Brain Tumor Segmentation (BraTS) dataset [74] is a neuro-oncological dataset where the segmentation task is to localize the healthy tissue (background) and the tumour (foreground) in the image. The 3D networks are tested on an in-house dataset containing 524 subjects, where the task is to segment age-related white matter hyperintensities. The findings of the paper-based on testing are that a mildly imbalanced dataset appears to be well handled by most loss functions designed for unbalanced datasets, but when the level of imbalance is increased, the loss functions based on overlap measure appeared to be the superior choice.

3.3.2 Combo loss

Taghanaki et al. [104] is presenting a method of using combination of loss functions to handle the class imbalance issue. They compare the cross entropy loss and the Dice loss [75] mentioned in section 3.2.2.

The proposed method uses a combination of the binary cross-entropy function and the Dice loss function. The binary cross-entropy function is a multi-class implementation of binary cross-entropy, where the cross-entropy is calculated for each class N. The proposed loss function is given by equation 3.6, where N is the number of segment classes times the number of batch samples, K the number of classes, and S. They use a parameter, α, to control the Dice and cross-entropy contribution to the loss, and a parameter, $\beta \in [0, 1]$, to control the penalization of false positives/negatives. When the β parameter is set to lower than 0.5, the false positive is penalized more than the false negatives.

The main advantage of the combo loss function is that the cross-entropy term enforces a trade-off between the false positives and false negatives, and simultaneously avoids getting stuck in a local minimum, as it also leverages the Dice term.

$$\begin{aligned} L = \alpha \left(-\frac{1}{N} \sum_{i=1}^{N} \beta (t_i - lnp_i) + (1-\beta)[(1-t_i)] \right) \\ -(-\alpha) \sum_{i=1}^{K} \left(\frac{2\sum_{i=1}^{N} p_i t_i + S}{\sum_{i=1}^{N} p_i + \sum_{i=1}^{N} t_i + S} \right) \end{aligned} \tag{3.6}$$

The loss function was tested on different datasets including a PET dataset of whole body volumes, and a MRI dataset for prostate segmentation, and an ultrasound dataset for left ventricular myocardial segmentation. A grid search was performed to find the best parameter settings for α and β. It found that setting the $\alpha = 0.5$, giving equal contribution to the dice and the crossentropy functions, was optimal. The optimal β parameters were found to be $\beta = 0.4$ for the PET dataset models, $\beta = 0.6$ for the MRI dataset models and $\beta = 0.7$ for the ultrasound images.

Their findings show that the proposed combo loss function outperforms all tested architecture with its associated loss function with 57% ± 24%, 38% ± 18%, 86% ± 5% in Jaccard, Dice and false positive rate (FPR), respectively for the PET dataset. For the MRI segmentation, the combo loss function improved the 3D U-Net and the 3D V-Net performance by 4.6% and 1.13%, and 43.8% and 16.7% in false negative rate (FNR). The ultrasound segmentation was improved by 8.23% and 3.4% in Dice, and 33.3% and 16.7% in FNR.

3.4 Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials

Krähenbühl et. al [60] released a paper in 2012 in which they introduced an inference algorithm for fully connected CRFs, also known as DCRF. The DCRF inference algorithm was proved to be useful for refining segmentation maps produced by FCNs [4, 117, 15].

The DCRF defines pairwise edge potentials by a linear combination of Gaussian kernels in an arbitrary feature space. The iterative DCRF optimization is carried out by minimizing the Gibbs energy, defined by the following equation:

$$E(x) = \sum_{i} \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j) \tag{3.7}$$

where ψ_u is the unary potential computed independently for each pixel or voxel by a dense segmentation model which outputs a label assignment distribution x_i. $\psi_p(x_i, x_j$ is the pairwise potential that is given by the following equation:

$$\psi(x_i, x_j) = \mu(x_i, x_j) \underbrace{w^{(m)} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j)}_{k(\mathbf{f}_i, \mathbf{f}_j)} \tag{3.8}$$

where $k^{(m)}(\mathbf{f}_i, \mathbf{f}_j)$ is a Gaussian kernel, $\mathbf{f}_i$ and $\mathbf{f}_j$ are the feature vectors for the pixels or voxels i and j. $w^{(m)}$ is a linear combination of weights, μ is a label compatibility function by the Potts model, $\mu(x_i, x_j) = [x_i \; x_j]$. This function penalizes nearby pixels or voxels that are assigned different labels.

Contrast-sensitive two-kernel potentials are used for multi-class segmentation and labelling, and are given by pairwise potentials with the following equation:

$$k(f_i, f_j) = w^{(i)} \underbrace{exp\left(-\frac{|p_i - p_j|^2}{2\theta_\alpha^2} - \frac{|I_i - I_j|^2}{2\theta_\beta^2}\right)}_{appearance\ kernel} + w^{(2)} \underbrace{exp\left(-\frac{|p_i - p_j|^2}{2\theta_\gamma^2}\right)}_{smoothness\ kernel} \tag{3.9}$$

where I_i and I_j are color vectors, p_i and p_j are position vectors. The appearance kernel exploits the fact that nearby pixels or voxels with similar values are likely to be in the same segment class. The smoothness kernel, on the other hand, functions as a smoothing filter that removes small isolated islands and regions. Pairwise terms can be influenced by adjusting their weights; w^1 and w^2. The effective range of the terms can be controlled by the kernel widths, where θ_α and θ_β control the appearance kernel, and θ_γ controls the smoothness kernel.

The inference is carried out by a proposed efficient inference algorithm. The algorithm is an approximation algorithm based on mean-field approximation of the CRF distribution. This approximation is an iterative message-passing algorithm. The message passing is performed using Gaussian filtering in feature space, which enables highly efficient approximations for high-dimensional filtering. This reduces the complexity of the message passing from quadratic to linear. The result is an approximating inference algorithm for DCRF and is linear in the number of variables N and sub-linear in the number of edges in the model.

Chapter 4: Methods

This chapter presents and discusses the methods used for this thesis. Section 4.1 will discuss the provided data for this thesis. In section 4.2 some of the architectural choices for the segmentation methods used are discussed. Furthermore, section 4.4 will present a loss function which deals with the imbalanced dataset. Section 4.5 will discuss a proposed framework for segmentation of MRI images. Section 4.6 will discuss how the ML models are trained. Section 4.7 will discuss a graphical user interface (GUI) created to visually evaluate segmentation masks. Finally, section 4.8 will go over the software and hardware used to develop and run the proposed methods in this thesis.

4.1 Data

The data for this thesis was provided by Sunnmøre MR-Klinikk. To get data for the collaboration project volunteers were scanned by Sunnmøre MR-Klinikk. The data gathered did not contain any sensitive information about the volunteers or any data that could identify them. Since the project is an ongoing project, the data was provided incrementally throughout the semester. Because of this, much of the experimentation early on was done using very few samples. The limited data had an impact on some of the decisions, as it was hard to verify if the models, decisions, and experimentation early on would apply to new data as well.

The data was provided primarily in the Nifti format. Some of the files provided were in the Dicom format but were converted to Nifti, as it is easier to stick to a single format. Nifti was preferred because of the more straightforward one-file structure. The Nifti format also allowed for compression, meaning less space taken on the hard-drive.

For the contents of the data, MRI T1, PD and FS weighted volumes were provided. The volumes were all aligned to match the segmentation mask the best possible. The dimensions of the data provided were $275 \times 400 \times 400$. The voxel dimensions were (0.4mm, 0.4mm, 0.4mm). The segmented ground truth masks were manually segmented by Sunnmøre MR-Klinikk. The segmented classes include the bone, PCL, and the ACL. The segmented masks were provided in a separate file, and the final number of samples amounted to 17 unique knees, giving $17 \times 4 = 68$ total Nifti files.

4.1.1 Data augmentation

Since the dataset provided was somewhat small, and because it was unclear in the start of how large the dataset was going to be, data augmentation was thought to be necessary to diversify the data to avoid overfitting. As previously discussed in section 2.10 data augmentation is typically a good method for model regularization when the data is sparse.

When deciding the augmentation, some things need to be taken into consideration. Since the target for training the models is a segmentation mask, some of the augmentations have to be applied uniformly to both the input sample and the output samples. For augmentation functions that transform the spatial representation of the input sample, the augmentation is uniformly applied to both the input samples and the output samples. Furthermore, it has been taken into account how the segmentation models can be sensitive to certain types of augmentations. The augmentation method parameters are evaluated to produce plausible samples resembling real-world MRI images. It would be pointless to train the models on data deviating too much from the original data, as they will then learn features not present in real-world data. The augmentation methods used, and the assumptions made for deciding the methods and parameters will be further discussed in this section.

Elastic deformation

Elastic deformation [87, 2] mentioned in section 3.2.3 was used for augmenting the data. When using elastic deformation, it is important to set the parameters appropriately to achieve realistic new samples. Figure 4.1b show an example of how the elastic deformation affects a volume slice using inappropriate parameter settings.

Cropping

In the data provided the anatomical structures in the MRI images are represented in various areas for every volume. To exploit this fact, we can use random cropping to create new spatial representations of the MRI images. Random cropping will improve the robustness of the feature detectors.

The cropping is applied by removing a random number of slices from the borders of the volume. Then, the volume is resized back to the original volume shape. By doing this, the volume's spatial characteristics have changed, and the volume shape is conserved. The maximum number of slices cropped is limited to keep the semantics of the volume. Figure 4.1c illustrates an example of using the cropping augmentation.

Gaussian noise

The MRI images are naturally very noisy and grainy. To exploit this fact, we can alter the noise characteristics of the images by adding random noise. For this, a method for adding Gaussian noise was implemented. For the method implemented, the noise is only added to the volume where the voxels are not equal to zero. By not adding noise to the background voxels of the volume will make the ML models better at distinguishing the background from the rest of the volume. The Gaussian noise should not be used with Gaussian blur, as the blur function will cancel out the noise.

The noise is added by adding an array with equal size of the original volume, where each element in the array is sampled from a normal distribution. Before the array is added, the array is element-wise multiplied with an array, cancelling out the noise otherwise added to the background voxels. Equation 4.1 shows the general equation for the Gaussian noise function. Where Z is the volume with voxels, x. nd is the normal distribution with its parameters, mean μ, and standard deviation σ. The Gaussian noise function is demonstrated in figure 4.1d.

$$Z'_x = Z_x + nd(\mu, \sigma) \forall x \in \mathbb{N}^n : Z_x \neq 0 \tag{4.1}$$

Gaussian blur

The Gaussian blur function is similar to the Gaussian noise function and can be beneficial as an augmentation method. Blurring out some of the noise will change the noise characteristics of the volume voxels. However, the blurring needs to be constrained, as the fine-scaled image edges and details can be blurred out by excessive blurring. Figure 4.1e shows how excessive blurring affects a volume.

By applying Gaussian blur to the volume, we are just convolving the volume with a Gaussian function. The Gaussian function for n dimensions is given by equation 4.2, where σ is the standard deviation for the Gaussian kernel.

$$G(Z) = \left(\frac{1}{\sqrt{2}\pi\sigma}\right)^n e^{\frac{\sum_{i=1}^{n}(x_i)^2}{2\sigma^2}}, \qquad Z \in \mathbb{R}^n \tag{4.2}$$

Brightness

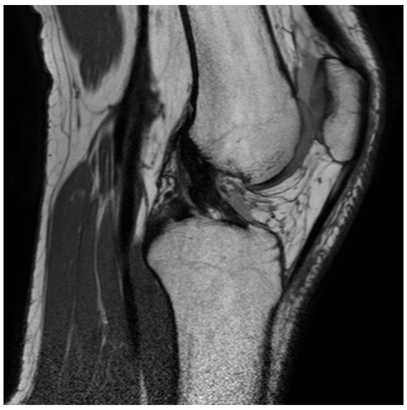
The MRI image brightness can vary to some degree. For augmentation, this can be exploited by slight changes in the brightness factor of the images. The brightness is changed by a multiplication of the volume Z with a factor C. However, because the brightness is changed uniformly over the volume the values are clipped between the minimum Z_{min} and maximum Z_{max} values of the volume, otherwise the values will be equivalent to the original volume when normalized. The clipping is performed by using max and min functions for every voxel in the volume Z.

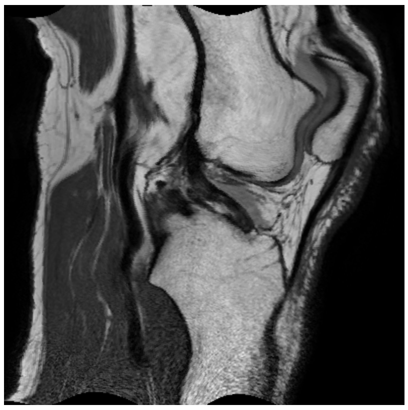
$$Z^{'} = max(Z_{min},\;\; min(Z \times C,\;\; Z_{max}))\;\; , \qquad C \in \mathbb{R} \tag{4.3}$$

Contrast

Similar to brightness, the image contrast can also vary to some degree in MRI images. Contrast augmentation is applied by using a factor C to control the contrast strength. Clipping is applied to avoid the previously mentioned normalization problem. The formula for the contrast implementation is given by equation 4.4, where $Z_{mid} = (Z_{min} + Z_{max}) \div 2$.

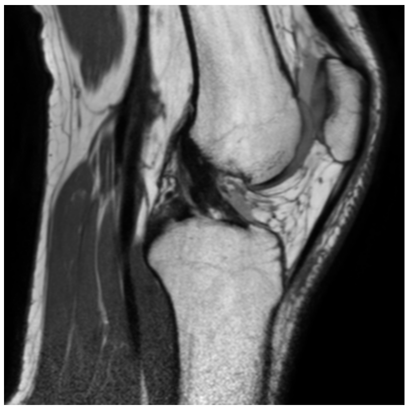
$$Z^{'} = max(Z_{min},\;\; min(Z \times C - Z_{mid} \times C + Z_{mid},\;\; Z_{max}))\;\; , \qquad C \in \mathbb{R} \tag{4.4}$$

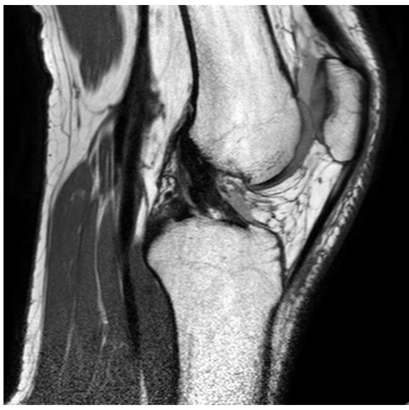

(a) Non-augmented


(b) Elastic deformation

(c) Cropping

(d) Gaussian noise


(e) Gaussian blur


(f) Brightness

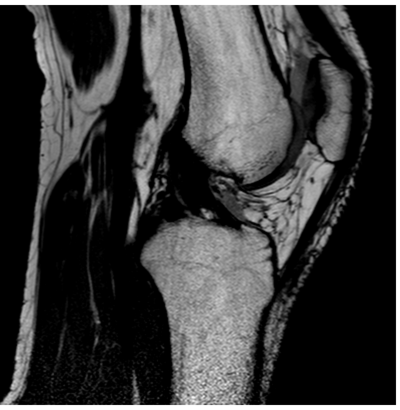

(g) Contrast

Figure 4.1: Examples illustrating the implemented augmentation methods using improper parameter values to visualize how the augmentation method will alter the original volume. Where (a) show the input sample, (b) elastic deformation, (c) cropping, (d) Gaussian noise, (e) Gaussian blur, (f) brightness, and (g) contrast.

4.1.2 Data generation

The augmentation functions were performed on the data with random parameters within a range. The maximum values of the ranges were controlled to avoid implausible samples. All the augmentation functions were implemented as a single function except the Gaussian noise and blur functions, which are combined to one function where activating only one of them by random choice with equal probability.

The augmentation process was done before training, creating an augmented dataset, also known as offline augmentation. The augmentation was primarily done on beforehand because some of the augmentation methods were quite computationally demanding and slowed down the training process.

The T1, FS, PD, and the segmentation masks files were paired, and 32 new augmented sets were created for each mask pair, giving a total of 1280 files. For each pair in a set, the augmentation functions were applied uniformly. Instead of applying all augmentation functions to all the new samples, the functions are randomly activated with a 75% chance of being activated. The function order is also randomly shuffled for each new sample. By shuffling the functions and randomly activating the functions, we get a more diverse dataset, with various combinations of augmentation. The following algorithm further explains the generation of new data:

Algorithm 1 Data generation

```
$n\_samples = 32$
create a set list containing uniform resource locator (URL)s for each label and its corresponding weighted images as a set
**for** Each training set in set list **do**
    load each volume in the set
    **for** $i = 0; i < n\_samples; i++$ **do**
        shuffle augmentation function list
        **for** Each function in augmentation function list **do**
            **if** $random(0, 1) < 0.75$ **then**
                perform augmentation function on whole set
            **end if**
        **end for**
        save the new augmented set
    **end for**
**end for**
```

4.2 Convolutional neural network architecture

When comparing the FCN architectures in section 3.2 the architectures are very similar in performance and structure. Either one would probably be adequate for the segmentation task for this thesis. A greater understanding of dealing with the limited dataset and the class imbalance of the data must be established. As the architectures are quite similar, if the problem can be solved by one algorithm, it can probably be solved by the other ones.

For this thesis, it was decided to focus primarily on the U-Net [87] architecture. The network is well established, especially for medical images, and has a lot of variants and literature around it. The network got a simple and easily understandable structure, does not require a lot of training time and data, and is fast enough for the problem at hand. The network can also work as a baseline if other networks were to be implemented.

U-Net

For this thesis, a 3D implementation of the U-Net was implemented, where 2D network operations are replaced by the corresponding 3D operations. In the U-Net paper [87] the proposed network did not use padding for the convolutional layers, leading to a reduced output resolution of the network. This method is more suitable for a patch-based segmentation method where the network is trained to segment the data patch by patch. The implementation of the U-Net for this thesis is using the "same" padding parameters for the convolutional layers, which results in the output segmentation map having the same resolution as the input volume resolution. Since the segmentation task for this problem is a multi-class problem, a softmax activation function is used for the final layer yielding voxel-wise probabilistic segmentation maps.

4.3 3D Dense Conditional Random Fields

Although the ML models are quite good at segmenting the volumes, they can still produce segmentation maps with irregularities, including holes and small isolated islands. To smooth and further improve the label assignment, a 3D DCRF was used as a post-processing function to fine-tune and improve the segmentation maps.

Given a segmentation model and an input volume to segment, the 3D DCRF inference process defines a maximum a posteriori (MAP) over the 3D volume. The segmentation map produced by the segmentation model is used to generate the unary potential for the DCRF. The input is used to calculate the pairwise potentials on the pairs of voxels. Finally, the inference is carried out by the inference algorithm.

When deciding the parameters for the DCRF the number of inference iterations was set to 15, as the inference process of the DCRF model was quite slow. Simple grid searches were performed to find the optimal parameters for the pairwise potentials.

4.4 Loss function for class imbalance

To find the best loss function for the problem the segment class imbalance was taken into account. The imbalance ratio is calculated as the percentage of voxels the segment class occupies in the total volume. The class imbalance ratio and standard deviation (STD) are presented in the following table, where **Non-BG** is the non-background classes combined. Table 4.1 show that the class imbalance is quite heavy for both the PCL and ACL classes. To deal with the class imbalance it was deemed necessary for a loss function that penalized errors for the smaller segment classes to a greater extent.

	Background	**Bone**	**PCL**	**ACL**	**Non-BG**
Imbalance ratio	0.89456	0.10413	0.00088	0.000424	0.10544
STD	0.00155	0.00154	1.36501e-05	9.87766e-06	0.00157

Table 4.1: Table showing the class imbalance ratio for the different segment classes.

For this thesis, three loss functions were implemented, including a weighted categorical crossentropy loss (WCCL), weighted Dice loss (WDL) and a weighted Jaccard loss (WJL). Inspired by the combo loss [104] discussed in section 3.3 a function for combination of loss functions were also implemented.

The WCCL function evaluates the class predictions for each voxel vector individually and averages over all voxels. By weighting the loss for each output channel, we counteract the class imbalance present. The weighted categorical cross-entropy loss function is given by the following equation:

$$WCCL = 1 - \sum_{c=1}^{C} w_c \sum_{n=1}^{N} ln(p_{cn})t_{cn} \tag{4.5}$$

where C is the number of segment classes, w is the weight for the segment class $_c$. t is the ground truth mask, and p is the predicted mask for the voxels $_n$.

The WDL function is similar to the GDL [100] function discussed in section 3.3, however as will be discussed further below it differs in how it is weighted. The WDL function will account for global spatial information and overlap of the compared segmentation masks. The WDL function is given by the following equation:

$$WDL = 1 - \sum_{c=1}^{C} 2w_c \sum_{n=1}^{N} \frac{t_{cn} p_{cn}}{t_{cn} + p_{cn}} \tag{4.6}$$

Similar to the WDL function a general WJL function were also implemented to compare the two. Similarly, the WJL function will also account for global spatial information and overlap of the compared segmentation masks. The WJL function is given by the following equation.

$$WJL = 1 - \sum_{c=1}^{C} w_c \sum_{n=1}^{N} \frac{t_{cn} p_{cn}}{t_{cn} + p_{cn} - t_{cn} p_{cn}} \tag{4.7}$$

All of the loss functions mentioned above use the weighting given by equation 4.8, where V_c is the size of the segment class in volume $_i$, and V is the size of the whole volume. N is the number of volumes in the training set. The weights are calculated by taking the average volume size for the segment classes and subtract it from 1, and is just the imbalance ratio in table 4.1 subtracted from 1.

$$w_c = \frac{1}{N} \sum_{i=1}^{N} \frac{1 - V_{ci}}{V_i} \tag{4.8}$$

When weighting the loss functions the weights are calculated on beforehand instead of calculating the weights while training. This weighting scheme reduces some of the computation done during training. It could be argued that it would be more accurate to calculate the weights for each volume, but given how relatively small variance of the volume sizes of the segment classes, it should not be a problem.

A function for combining an arbitrary number of loss functions was implemented. The motivation of combining loss functions was to combine the WCCL with one of the overlap loss functions. By combining both cross-entropy and each voxel position, deviation in the predicted map is penalized, and at the same time exploit the global spatial information provided by the use of a loss function penalizing similarity error.

The combination of loss functions is done by using equation 4.9. Where E_j is the loss function for K loss functions, and v_j denotes how much the loss function is weighted.

$$E = \frac{1}{NK} \sum_{i=1}^{N} \sum_{j=1}^{K} E_{ij}(t_{ij}, p_{ij}) v_{ij} \tag{4.9}$$

4.5 MRI segmentation configuration platform

A lightweight platform for the automatic segmentation of MRI was created. The platform is inspired by NiftyNet [33, 34], which is an open-source CNN platform for research in medical image analysis and image-guided therapy. However, NiftyNet was limiting in some concerns, and it lacked some features and configurations that were wanted for this thesis.

The primary motivation of the platform was to make for quicker training, testing, and validation of models. The platform use configurations files to set up the parameters for the different models. By using file configurations for the models, we can automate away much hard-coding of parameters, and model initialization. Another advantage is how configurations for models can be created and changed even if the program is compiled to an application, which is useful if models are to be used in real-world applications.

The platforms file structure is built on the YAML Ain't Markup Language (YAML) data-serialization language [83]. The reason for choosing YAML is that it is readable, has good support for the Python programming language and uses similar nesting structures and formats to Python. Comparing YAML to other data-serialization languages as JavaScript Object Notation (JSON) [22], it was found that YAML was a better fit for this project as the nested structures makes for more neat and minimalist configuration files, and it utilizes more advanced data formats than simple initialization (INI) file parsing.

For the platform, there are four different model configuration structures. These structures include a structure for machine learning models, a structure for DCRF models, a configuration structure for ensemble averaging models and a structure for assembling different models. Common for all of the configuration structure is parameters for the data transformation, which includes resizing, cropping, and padding.

The configuration file structure for machine learning models contains a field for which network to use, fields for network architecture parameters, fields for the network training parameters, and more general fields like saving locations for network weights. The configuration file structure can easily be used for both training and inference, as the structure contains all parameters necessary for setting up the network for both use-cases.

The DCRF file structure is made for carrying out DCRF inference over the segmentation map output from a model. The structure contains fields for the parameters of the DCRF process. Further, a field specifying the model which will be used for the inference process. The model configuration can be directly written inside of the field or a URL pointing to a model configuration file that can be provided.

The ensemble model configuration structure is made for combining and averaging the outputs of multiple models. Ensemble averaging is quite useful for the segmentation problem of this thesis because we are provided with three different types of weighted MRI images. By training different models on the different weighted MRI images, we can utilize all the different representations of the knee to get a more robust output as the errors made by the different models will average out.

To ensemble different models, a list field is used to specify all the models to average in the ensemble process. It is made possible to specify the whole model configuration as text inside this list or use the preferred method of adding a URL to point to a model configuration to include in the ensemble process. By separating the model configuration files like this, it is easy to add and remove models from the ensemble process.

When using the ensemble model configuration for inference, all the models of the ensemble produce a segmentation map by inference given its input. Further, the ensemble models output map is given by the average of the segmentation maps. It is also possible to weight the contribution of each model by providing a weight list in the configuration file containing the weights for each model in the ensemble. The output segmentation map is then given by equation 4.10, where y is the output segmentation map, N is the number of models, f is the models, x is the input volumes, and w is the weights.

$$y = \frac{1}{N} \sum_{i=1}^{N} f_i(x_i) w_i \tag{4.10}$$

The assemble configuration structure is made for assembling different models trained to segment individual classes. This assembly process is illustrated in figure 4.2. The models to assemble is defined in a list, and similarly to the ensemble structure, it is possible to write whole model configurations inside of the list or point to URLs for the models to assemble.

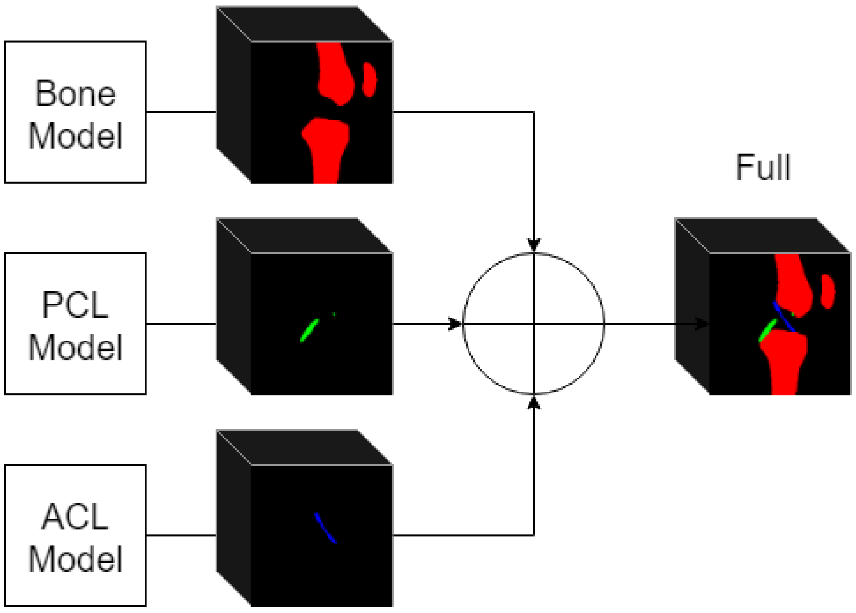


Figure 4.2: Assemble model

The assembly process is carried out by combining the segmentation maps produced by the different models into one segmentation map. However, because our models produce softmax predictions, it is not as simple as summing the segmentation maps together, because if doing so, overlapping background regions will yield a higher probability for the background segment class.

The segmentation maps for each model is used to produce a *background region mask* by using the argmax function over the map. This mask will be used to determine where the background labels are. Furthermore, we take the segmentation maps and set the background class value to zero. Then we sum the segmentation maps together. Finally, we use the background region mask to determine the background area in the segmentation map. This process is further explained in the following algorithm:

Algorithm 2 Model assemble

```
get number of models $\rightarrow$ N
initialize segmentation map list
**for** each inference model **do**
    get segmentation map from inference model given its input data
    add segmentation map to segmentation map list
**end for**
sum maps in segmentation map list $\rightarrow$ summed segmentation map
$argmax$(summed segmentation map) $\rightarrow$ background region mask
set all background confidence labels of the segmentation maps in segmentation map list to zero
sum all maps in segmentation map list $\rightarrow$ final segmentation map
set background confidence labels of the final segmentation map given indices to N
return $argmax$(final segmentation map)
```

4.5.1 Model pipeline

The segmentation platform is made flexible as it is possible to create models following different pipelines. When loading a configuration structure for inference, the hierarchy of structures will be automatically configured to a single model, and makes it possible to configure different structures arbitrarily, and allows for a diverse range of different model pipelines.

As an example, one of the proposed models makes use of all of the configuration structures. This model is quite extensive, and will be further detailed in section 5.4 and section 6.6. The pipeline is illustrated in figure 4.3. For this pipeline, The segmentation task is split up into three averaging ensembles built of ML models, for the three-segment classes. The output of all three ensembles are further assembled by an assembler, and finally, the assembled segmentation map is refined and smoothed by a DCRF inference process.

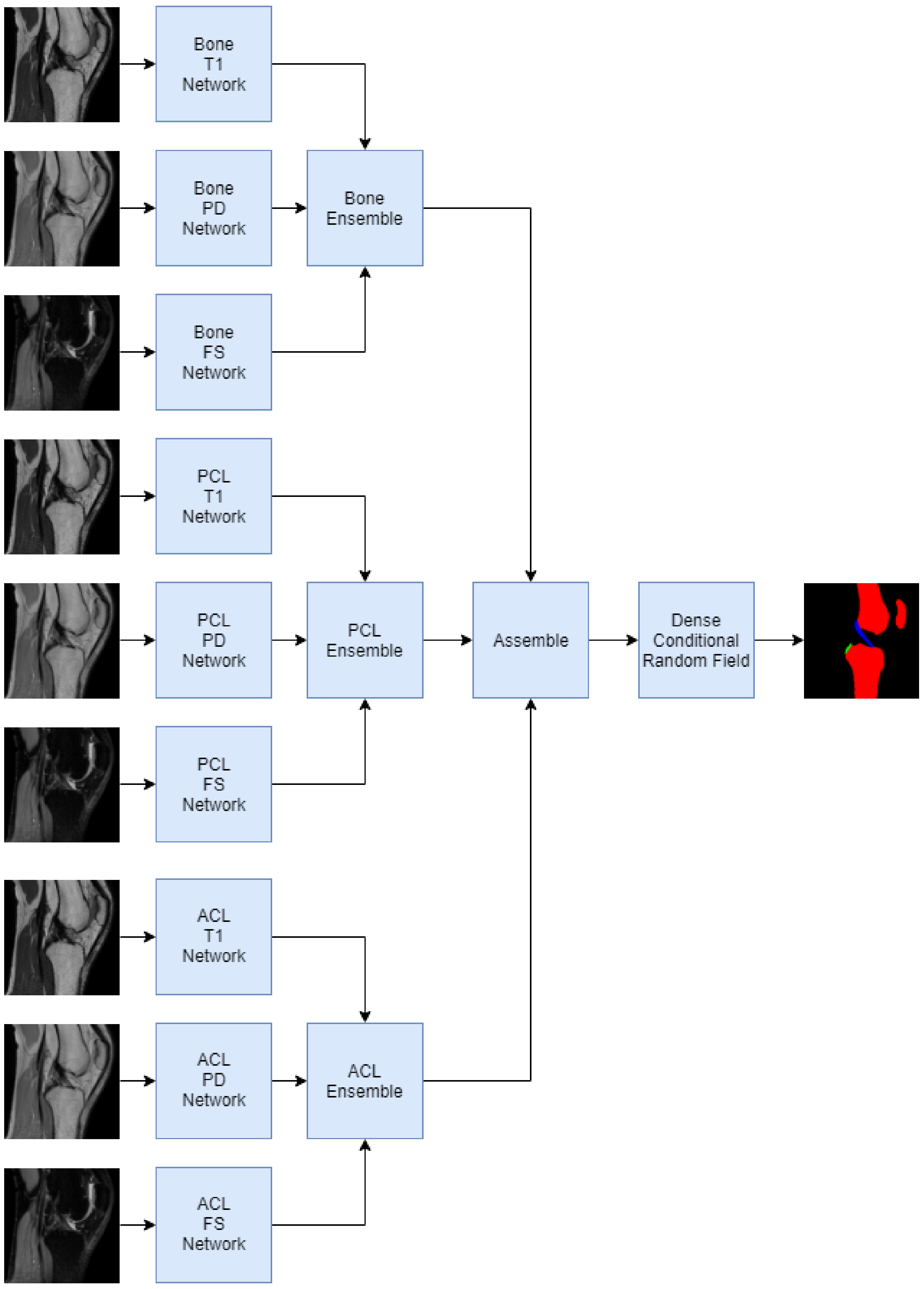


Figure 4.3: Extensive model pipeline

4.6 Training

For training, it is typical to do an 80-20 split of data, where some 80% of the data is used for training, and 20% is used for validation and testing. The initial split was ten images for training and 3 for testing. However, some of the data was given very late, and the final split was using ten samples for training and 7 for validation and testing, resulting in a split closer to 60-40. Because of time limitations, the experiments were not re-done with a new split closer to 80-20. However, it could be argued that by splitting 60-40, we can comprehend how well the model performs given very little training data. It is common to split the testing set to a validation set and a testing set, but because of the limited amount of data, the testing set was also used as a validation set. The set was used for early stopping and was also used to control for overfitting. However, the results from the training validation will be different from the testing results. When the models are tested, the output segmentation masks are re-scaled to its original resolution, which differs from the validation testing where the validation metric is calculated using the model's output resolution. The validation will also validate the soft labels and not the final segmentation mask.

The models were trained using a batch size of one to reduce memory consumption. Smaller batch sizes have also been shown to improve generalization [57, 42], and it removes the need for batch normalization layers [49], which have been found to reduce the performance of U-nets in some cases [16].

Further, the only pre-processing used on the input volumes was normalization. The input volumes were normalized between zero and one. Normalization is done to get all the data on the same scale, as it is a problem if all the input volumes have different feature scaling, which can hurt the training. For the ground truth segmentation masks, the mask is converted to a segmentation map by using one-hot encoding as discussed in section 2.4.2.

Because of the limited dataset, the Adam optimizer was used for training the models. As discussed in section 2.5.2 the Adam optimizer is an adaptive learning method that is advantageous for a limited dataset. The parameters set for the optimizer follows those provided in the original paper [58].

Because of memory limitations for the computer system used for training the machine learning models, it was not possible to load all the MRI volumes into memory simultaneously. The Keras [14] library already has data-generators for images useful for large datasets, but is limited to 2D-image formats such as portable network graphics (PNG) and joint photographic experts group (JPEG). Dealing with a dataset of Nifti files made it necessary to create data-generators for training the models. Two similar data-generators were created for feeding 3D volumes or 2D slices to the neural networks.

4.7 Segmentation mask comparison tool

For this thesis evaluating the segmentation masks by inspection was crucial, as using the segmentation metrics alone was not sufficient by itself to determine if a segmentation mask is good or not. Because of this, a simple segmentation mask comparison tool was made. The proposed tool only concerns inspection and comparison of MRI segmentation masks and is tailored to the specific needs for this thesis. Further, this made the inspection process of the segmentation masks substantially more effective. The core use case for the proposed tool was to compare the ground truth and predicted segmentation masks of volumes.

The GUI was split into three tabs. The first tab is for comparing the segmentation masks in slices. The 2D tab consists of a grid of 8 image windows. These image windows include various tools to control the content of the windows. A slider is added to slide through the slices of the volume. It is possible to select which anatomical plane to slice through and present in the window. Further, a camera tool is added to export the current inspected slice of the window as an image file.

The image windows are established to enable the use of the real/raw MRI volume as the background, and segmentation masks as an overlay. We can then see how well the segmentation mask aligns with the corresponding tissues in the real volume. Combo boxes are added to change the background and overlay volumes. Furthermore, a slider is added to change the opacity of the overlay.

A 3D tab is added to generate and render 3D models from the segmentation volumes. A 3D representation of the segmentation masks is quite useful, as it can give a quick overview of how well the segmentation masks compare without having to slide through them and inspect each slice. Two rendering windows are added to compare two segmentation masks in 3D simultaneously. The segmentation masks are visualized in 3D by generating a mesh from the masks using the marching cubes algorithm, which is discussed in section 2.14. Because the output mesh is somewhat "blocky" when using the marching cube algorithm, a button is added which toggles a smoothing function on the mesh.

Two buttons for controlling the camera of the rendering windows are added. One for resetting the camera to the original view and another to copy the camera transform from the other render window. Finally, a button for exporting the rendered 3D model to a 3D file is added. The 3D model can be exported as a ".obj" file which can be used in 3D game engines and modelling software, ".vtk" file is usually used for scientific purposes or as a ".stl" file which can be used for 3D printing.

The last tab of the segmentation comparison tool is for metric evaluation of the segmentation masks. This tab is intended to compare the overlay error and pixel accuracy of two selected segmentation masks. Further, a plot widget is added to compare the segmentation accuracy of two segmentation masks for each slice through the volume. We can then easily see where the segmentation mask has the most errors, and we get the insight of where the segmentation models are producing the most errors. Another feature of the plot widget is the clickable ticks along the plot curve. Clicking a tick brings up a window illustrating the compared volume slices corresponding to the tick's x position. We can then quickly compare and inspect what types of errors lead to a lower accuracy for a particular slice.

In addition to the plotting window, a metric list is added listing various metrics calculated over the two selected segmentation masks. The metrics are calculated for the two segmentation masks are the same metrics listed in table 2.1 in section 2.9.4. The list allows for a quick overview of how similar the selected masks are numerically.

4.8 Software and hardware

When choosing the software tools and software libraries to use for the thesis, some criteria were set for choosing the correct software tools and software libraries. The criteria set for the software was how useful it was for the task at hand, prior experience, compatibility with other software, and documentation. The software used for this thesis is listed here:

- **Nibabel** [9] is a Python library for reading and writing standard neuroimaging file formats. The library was used extensively to load and save files in the nifti format.
- **Numpy** [53] is a Python library that adds support for multi-dimensional arrays and matrices. The library also includes much functionality for operating on these arrays. This library was used a substantial amount when editing the MRI volumes.
- **PyQt5** [19] is a Python binding library for the cross-platform GUI toolkit Qt. The library was used to build a segmentation comparison tool.
- **Visualization toolkit (VTK)** [93] is an open-source cross-platform library that provides developers with an extensive suite of software tools for 3D computer graphics, image processing, and visualization. The Python binding library was used to generate the 3D meshes from the segmentation masks for the comparison tool. The libraries PyQt widgets were also used for rendering and visualizing the 3D meshes.
- **PyQtGraph** [12] is a pure-python graphics and GUI. The library includes a lot of useful plotting widgets for the PyQt platform. The library was used for the plotting purposes for the comparison tool.
- **Matplotlib** [45] is a Python 2D plotting library which produces publication quality figures. The library was used for general plotting purposes throughout the thesis.
- **elasticdeform** [107] is pyton library which implements elastic grid-based deformations for N-dimensional images which is based on [87].
- **Pydense** [5] is a wrapper library based on [60]. The library was used for 3D DCRF inference.

- **Keras** [14] high-level neural networks API, written in Python. Keras is capable of using different machine learning libraries as backend, and for this thesis, Tensorflow [105] is used as backend. Tensorflow is an end-to-end open-source platform for machine learning. The combination of Keras using TensorFlow as a backend was used for the machine learning part of this thesis.
- **draw.io** [71] is an open platform where you can create and share diagrams. The software was used to draw various figures and illustrations for the thesis.

For training the machine learning models for this thesis an Nvidia RTX 2080 Ti GPU were used, and most of the testing was performed using an Nvidia GTX 1070 GPU. The final model evaluation and experiments were performed using a virtual machine on Azure cloud services.

Chapter 5: Experiments

This chapter will present the various experiments conducted. Section 5.1 gives details on an experiment testing if the data augmentation methods are working or not. Further an experiment testing various loss functions is described in section 5.2. In section 5.3 an experiment testing various input resolutions for the U-Net architecture is briefed. Finally, in section 5.4 describes an experiment fragmenting the segmentation task.

5.1 Data augmentation experiment

This experiment will examine how the established data augmentation methods affected the generalization of the machine learning models. The experiment is motivated by the assumption that a model trained on the augmented data will generalize equally well for a limited dataset. This hypothesis will be tested in this experiment.

The experiment is limited to model trained on the T1 MRI images. The experiment is performed using the 3D U-Net with a resolution of $128 \times 224 \times 224$ as the input size and $22^0 \rightarrow 22^4$ as the number of filters for the U-Net multiple stages. Because the augmented dataset contains 320 samples, the steps per epoch for the non-augmented dataset is set to the same size as the augmented dataset.

5.2 Loss function experiment

To find the best-suited loss function for dealing with the class imbalance present in the dataset a comparison experiment was conducted, testing various loss functions and combinations of loss functions.

The loss functions tested were the Dice loss [75] function, WDL, GDL [100], Jaccard loss, WJL and WCCL. Two combinations of loss functions were also tested, the WDL function combined with WCCL function and the combination of the WJL function with the WCCL function. The combo loss function [104] was not tested. Because they use a binary formulation for the multi-class problem their function could not be implemented directly for the U-Net implemented in this thesis.

A model configuration was made for each loss function. The experiment was conducted using the 3D U-Net architecture using $22^0 \rightarrow 22^4$ filters following the standard U-Nets multistage pattern. The input resolution was set to $124 \times 224 \times 224$.

The models were trained on the augmented T1 dataset. For training, the models were given 12 epochs to train using early stopping, with the patience parameter set to two. The patience parameter is the number of consecutive epochs with no improvement after which the training stops. This parameter is used to save time as it pointless training any further if further training does not lead to any further improvement.

5.3 Resolution experiment

A larger experiment was conducted, testing various input sizes for the 3D U-Net. The resolution of the segmentation mask is dependent on the resolution of the input. However, higher resolution input increases memory usage, thus limiting the feasible size of the model. Therefore, a good balance between input resolution and model size needs to be established. To optimize for accuracy the GPUs memory limitations were pushed to the limit.

To further test how well the 3D U-Net performs on the different weighted MRI images, a model for each type of weighted MRI images was created for each input resolution. By doing this, we gain insight into how the model will perform given the various weighted MRI images and is beneficial information for future work.

To test the average ensemble configuration structure an ensemble of models for each type of weighted MRI were created for the various input resolutions. The ensembles were also tested using the weighting parameter to see if it would yield an improvement over just averaging them equally.

For the training process the models were given 12 epochs to train, and early stopping with patience of two epochs was used. The models were trained on the augmented datasets using the combination of the WDL function and the WCCL function. Table 5.1 shows the resolutions tested as well as the number of filters used and the total number of parameters.

Resolution	**Filters**	**Parameters**
$64 \times 160 \times 160$	[64, 128, 256, 512, 1024]	90,292,868
$96 \times 192 \times 192$	[38, 76, 152, 304, 604]	31,833,896
$128 \times 224 \times 224$	[22, 44, 88, 176, 352]	10,671,368
$160 \times 256 \times 256$	[14, 28, 56, 112, 224]	4,322,168
$192 \times 288 \times 288$	[8, 16, 32, 64, 128]	1,411,796

Table 5.1: Resolution experiment parameters.

5.4 Fragmented segmentation experiment

An assemble-configuration was created to assemble the output from models trained for each class individually. This method is using a divide and conquer strategy and the segmentation task is then split up into three segmentation tasks, where each model will segment an individual class.

For the ACL and ACL segment classes the volumes are cropped to the area which the segment class usually occupy. The cropping parameters are extracted by finding the extreme points for each segment class. Since the segment classes reside in various locations for each data sample, the cropping parameters are decided by the global extreme-points for the whole dataset. Further, safety margins are added to the cropping parameters, to account for possible future data having lower or higher extreme points of the segment classes. The following table shows the cropping parameters and input resolutions:

	Sagittal crop	**Coronal crop**	**Axial crop**	**Input resolution**
Bone	None	None	None	$124 \times 224 \times 224$
PCL	$78 \rightarrow 174$	$117 \rightarrow 277$	$71 \rightarrow 263$	$96 \times 160 \times 192$
ACL	$95 \rightarrow 191$	$151 \rightarrow 279$	$85 \rightarrow 309$	$96 \times 128 \times 224$

Table 5.2: Fragmented model parameters

For each segment class an ensemble of models is created. The bone model is trained using the 3D U-Net using $22 \times 2^0 \rightarrow 22 \times 2^4$ as the filter parameters. As the input size of the ACL and PCL were smaller than the input size for the bone segment class the number of filters were set to $32 \times 2^0 \rightarrow 32 \times 2^4$. The models were trained using the combination of the loss functions WDL and WCCL on the augmented datasets.

The motivation for this experiment is that a single model might provide higher accuracy when the model's convolutional layer filters only have to be trained to segment a single class. Another advantage is the increased resolution of the segmentation masks, as the segmentation maps for at least the ACL and the PCL segment classes will be the same resolution as their original resolution. Due to the smaller input size, the model size can be increased without exceeding the memory limits, It also removes the need to down-sample the input volumes. The class imbalance also decreases, as a substantial amount of the background is cropped away.

Chapter 6: Results

This chapter will present and briefly discuss the results from the methods and experiments. The metrics used in this section is explained and discussed in section 2.9. Section 6.1 will showcase the segmentation mask evaluation tool. Section 6.2 presents the results from the data augmentation experiment. Section 6.3 presents the results from the loss function experiment. Section 6.4 presents the results from the resolution experiment. Section 6.6 presents and discusses the results for the use of DCRF as an post-processing filter for two of the segmentation models. Finally, 6.7 presents a visual inspection of the results and findings.

6.1 Segmentation mask evaluation tool

In this section, the segmentation comparison tools GUI will be presented. The images presented were taken on a desktop computer with a screen resolution of 1920×1080. Figure 6.1 illustrates the 2D visualization tab, showing an example. The top row shows the ground truth segmentation mask. The bottom row shows the predicted segmentation mask of the same knee. The first column shows the knee in the sagittal plane, the second column, in the coronal plane, the third column, in the axial plane, and the last the mask, without a background in the sagittal plane.

Figure 6.2 illustrates the 3D visualization tab showing a comparison of two smoothed 3D models generated by the same two segmentation masks illustrated in figure 6.1. The ground truth segmentation model is illustrated on the left, and the prediction model on the right.

The segmentation mask evaluation tab is illustrated in figure 6.3, and shows the evaluation of the two masks previously illustrated in figure 6.1 and 6.2. Finally, in figure 6.4 we see an example of the comparison window which appears when one of the ticks are clicked.

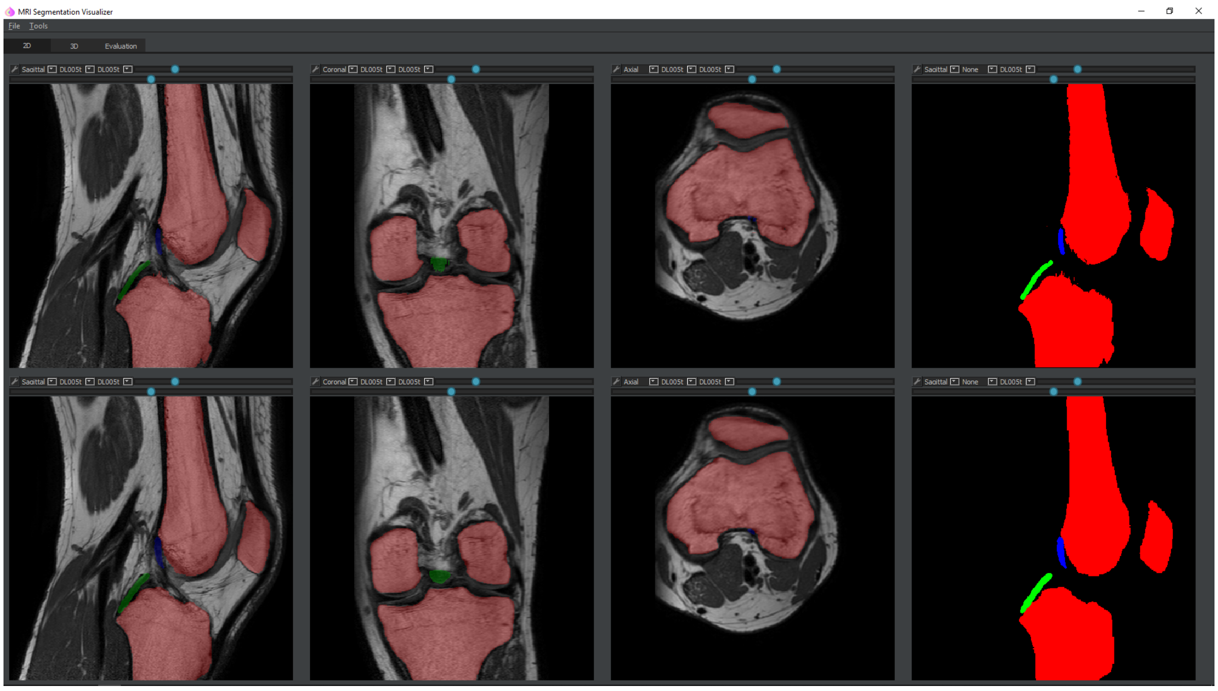


Figure 6.1: Image showing the tab for 2D visual comparison. This example show the ground truth slices on the top row, and prediction on the bottom row.

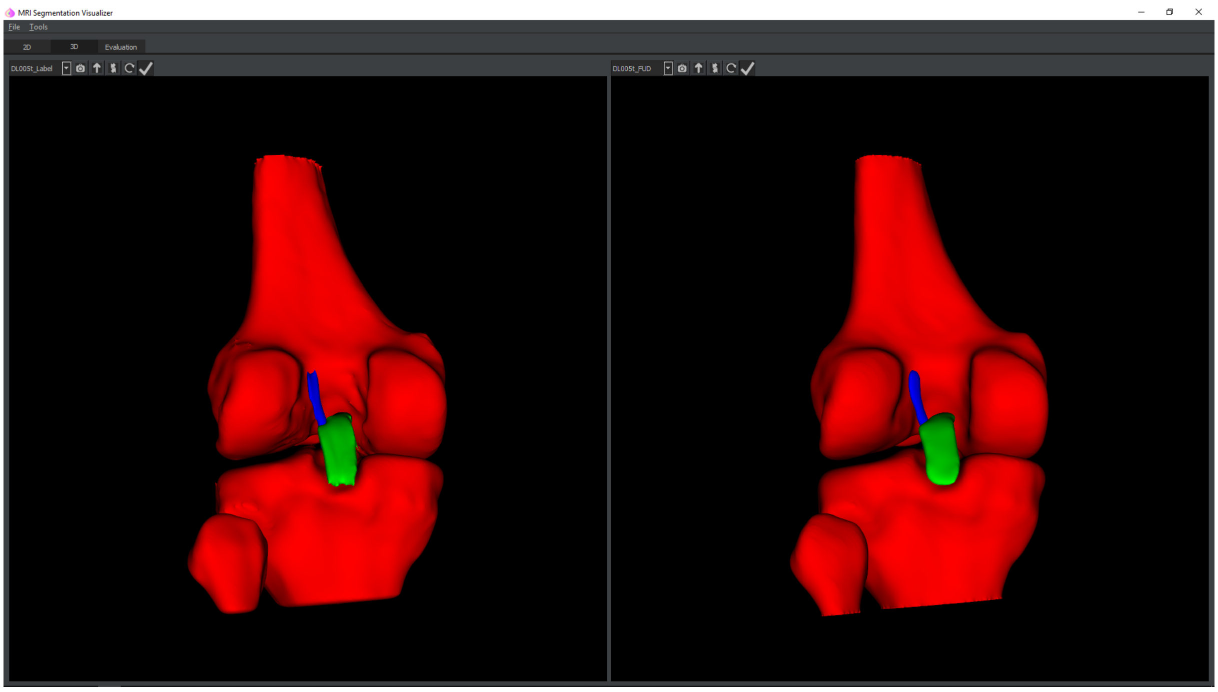
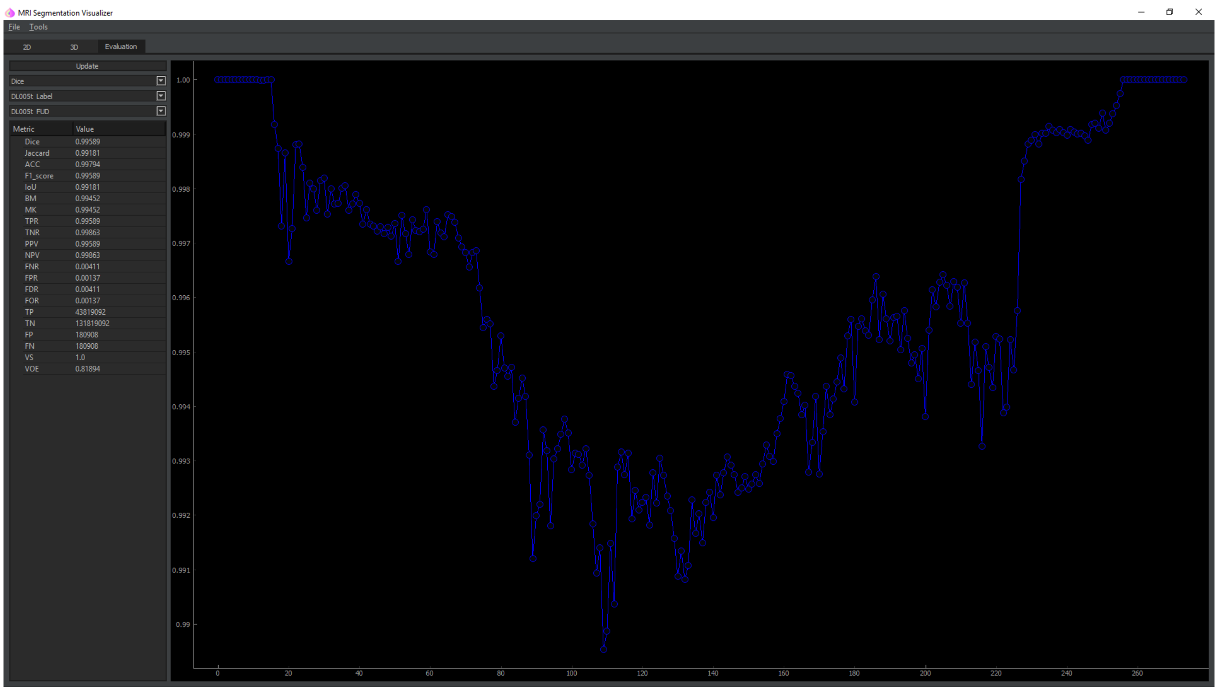
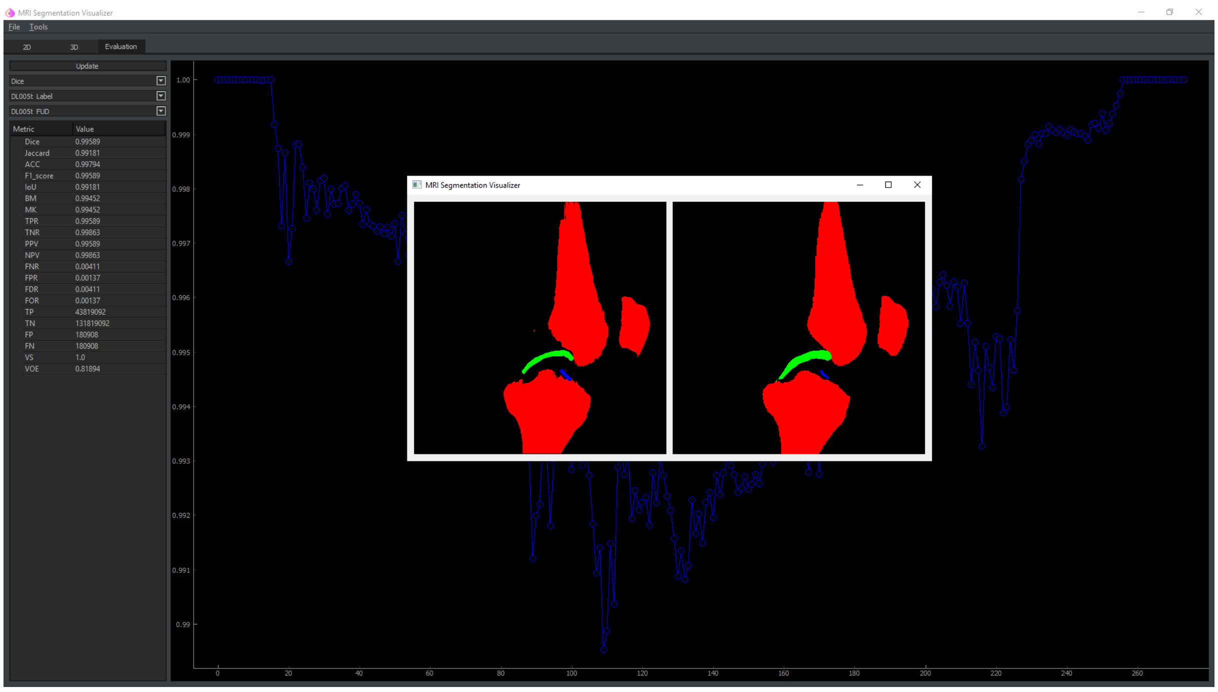


Figure 6.2: Image showing the tab for 3D visual comparison. This example show the ground truth on the left, and prediction on the right.



Metric	Value
Dice	0.99589
Jaccard	0.99181
ACC	0.99794
F1_score	0.99589
IoU	0.99181
BM	0.99452
MK	0.99452
TPR	0.99589
TNR	0.99863
PPV	0.99589
NPV	0.99863
FNR	0.00411
FPR	0.00137
FDR	0.00411
FOR	0.00137
TP	43819092
TN	131819092
FP	180908
FN	180908
VS	1.0
VOE	0.81894

Figure 6.3: Image showing the tab for evaluation of segmentation masks. The graph shows the Dice coefficient for every slice along the sagittal plane, comparing a ground truth mask with a prediction mask.



Metric	Value
Dice	0.99589
Jaccard	0.99181
ACC	0.99794
F1_score	0.99589
IoU	0.99181
BM	0.99452
MK	0.99452
TPR	0.99589
TNR	0.99863
PPV	0.99589
NPV	0.99863
FNR	0.00411
FPR	0.00137
FDR	0.00411
FOR	0.00137
TP	43819092
TN	131819092
FP	180908
FN	180908
VS	1.0
VOE	0.81894

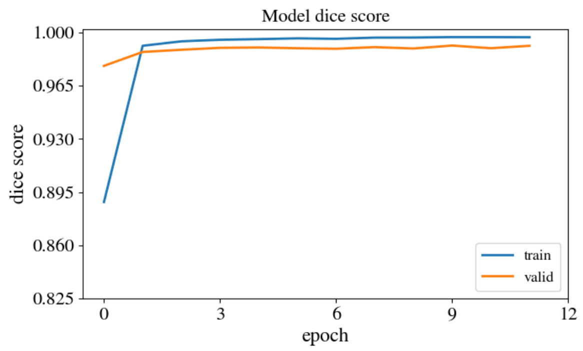
Figure 6.4: Image showing the comparison window when clicking a tick on the graph. This example show the ground truth on the left, and prediction on the right.

6.2 Data augmentation experiment

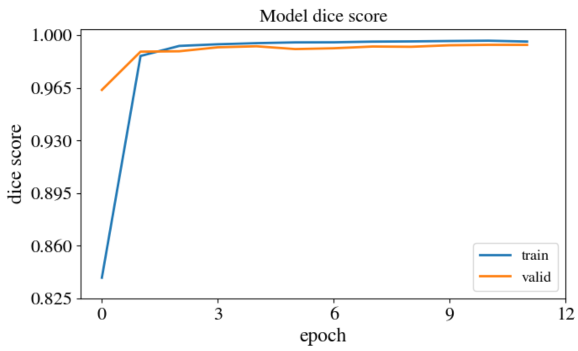
Figure 6.5 shows the graphs of how the Dice scores of the two models developed throughout 12 epochs. The graph in figure 6.5a illustrates the model trained on the non-augmented dataset, and figure 6.5b illustrates the graph for the model trained on the augmented dataset. The models converge after only three epochs. This also demonstrates the utility of early stopping, as it is pointless to train the model when it no longer improves. Table 6.5 the results from this experiment, where ± denotes STD.

	Accuracy	**Dice**	**Jaccard**
Augmented	0.99657 ± 0.00087	0.99314 ± 0.00173	0.98638 ± 0.00341
Non-Augmented	0.99505 ± 0.00256	0.99015 ± 0.00332	0.98045 ± 0.00716

Table 6.1: Accuracy comparison between models trained on augmented and non-augmented datasets.



(a) Model trained on non-augmented dataset



(b) Model trained on augmented dataset

Figure 6.5: Model Dice score development over 12 epochs for models trained on non-augmented and augmented datasets.

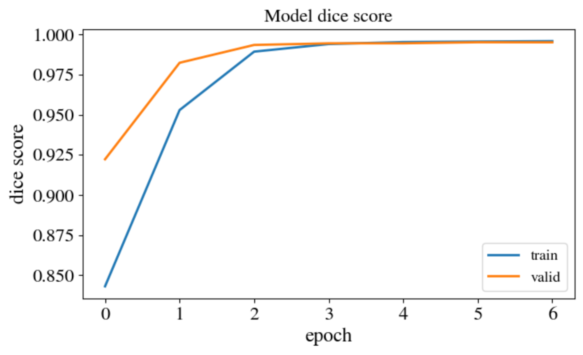
6.3 Loss function experiment

The models trained using the Dice loss, GDL, Jaccard loss, and the WJL function did not surpass a local minimum, and only returned blank segmentation masks consisting exclusively of background voxels. The same models also stopped after two to three epochs showing no improvement.

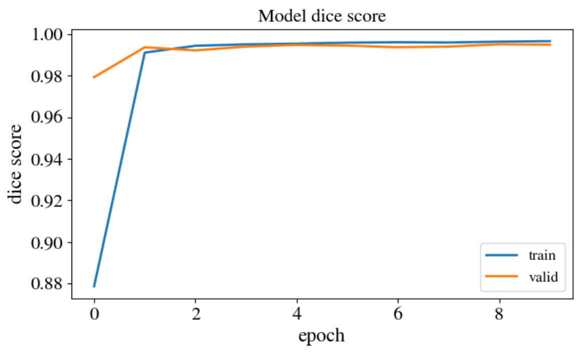
Figures 6.6a and 6.6b show that the model trained using a loss function combining a cross-entropy term and an overlapping term performed very similarly. Both models performed almost exactly the same, and the model trained with the WDL ended up with an average validation Dice score of 0.99507, compared to the model trained using the WJL function, which scored 0.99488.

As seen in figure 6.6c, the model trained on the WCCL loss function was quite unstable, evident from the highly fluctuating validation score. The training score is steadily climbing and has not plateaued yet. The training Dice score for the model trained on the WCCL function ended up at 0.92569, and the validation score at 0.95490. To investigate if the WCCL function would have performed better if given more epochs to train, the model was retrained, but initialized with the network-weights gathered from the model trained using the combination of WCE and weighted WDL as the loss function. The retrained model ended up with a validation score of 0.9946.

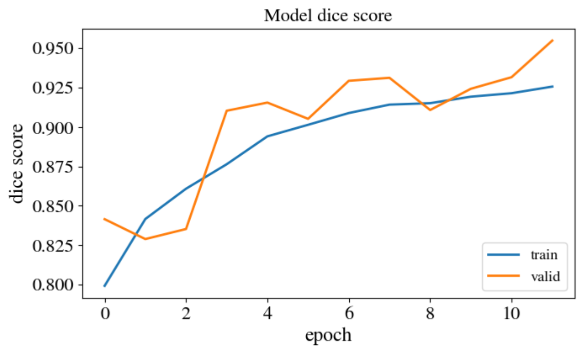
How the WDL function performed in this experiment can be seen in figure 6.6d. The WDL function plateaued and ended up on a validation Dice score of 0.99105, before it dipped down to 0.906475.



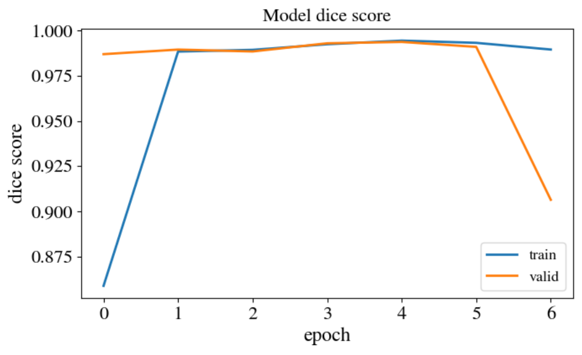
(a) WCCL and Weighted Dice



(b) WCCL and Weighted Jaccard



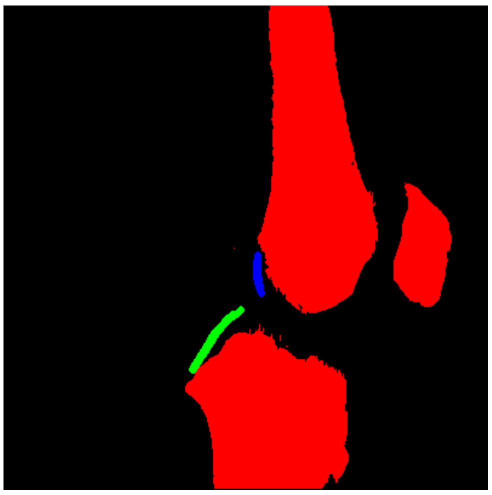
(c) WCCL



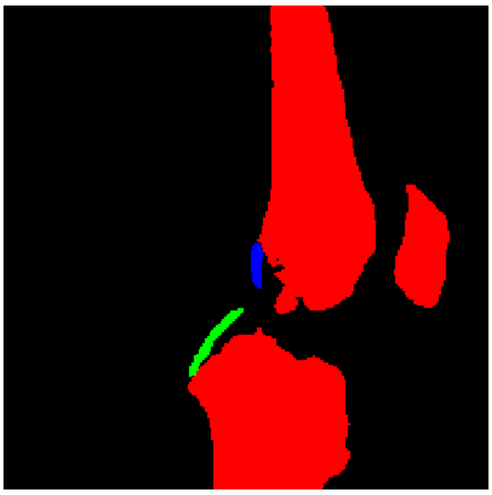
(d) Weighted Dice

Figure 6.6: Model Dice score development for models trained with different loss functions.

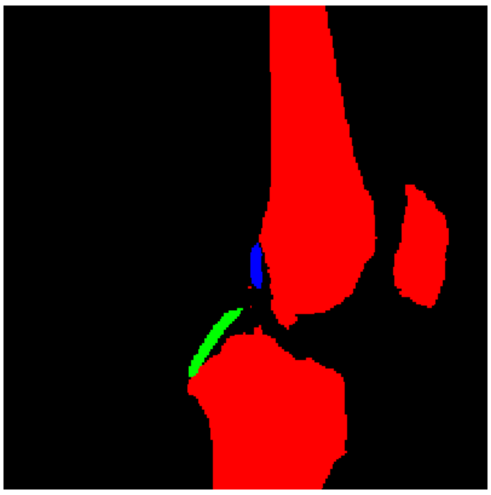
Figure 6.7 illustrates an example segmentation mask by the different models. From the figure, we see that the combo loss functions are producing plausible results when compared to the models trained on the WCCL and WDL functions.



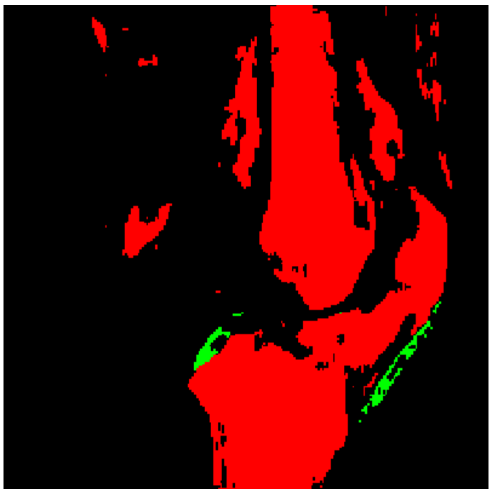
(a) Ground truth



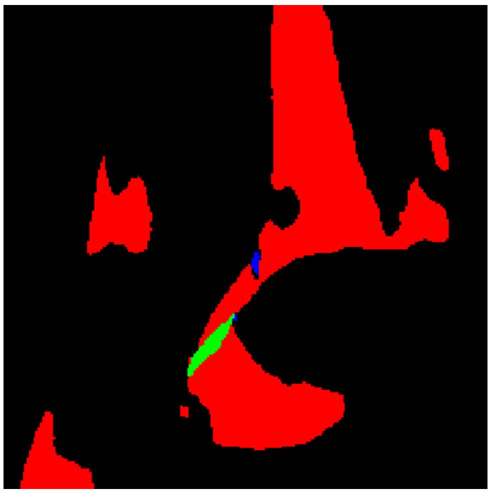
(b) WCCL and Weighted Dice



(c) WCCL and Weighted Jaccard



(d) WCCL



(e) Weighted Dice

Figure 6.7: Output mask from models trained on different loss functions compared to the ground truth.

6.4 Resolution experiment

In the tables 6.2, 6.3 and 6.4, the five different input resolutions tested in the experiment are numbered from 1 to 5. The ensemble is abbreviated to *ens* and the weighted ensemble is abbreviated to *wens*. Table 6.4 presents the results from all of the models trained for this experiment.

Table 6.2 show the average scores for the models trained on different weighted MRI and the ensemble models. From the table, it seems like the U-Net performs the worst when segmenting the FS weighted images, and performs the best with the T1 weighted images. The ensembles model accuracy seems to be weighted down by the FS and PD images scoring lower than the T1 images. Also, the weighted ensembles do not seem to improve the segmentation by much over the regular ensembles.

	Accuracy	**Dice**	**Jaccard**
fs	0.99153 ± 0.00076	0.98305 ± 0.00151	0.96686 ± 0.00286
pd	0.99268 ± 0.00261	0.98535 ± 0.00522	0.97123 ± 0.01006
t1	0.99651 ± 0.00014	0.99295 ± 0.00032	0.98615 ± 0.00054
ens	0.99601 ± 0.00022	0.99202 ± 0.00045	0.98418 ± 0.00088
wens	0.99603 ± 0.00022	0.99205 ± 0.00044	0.98425 ± 0.00086

Table 6.2: Average scores for the models trained on different weighted MRI images and ensemble models.

Table 6.3 show the average scores for the models trained using the different input resolutions. It seems like a middle ground between input resolution and model parameters is the obvious choice. However, we do see that the highest input resolution yielded the highest scoring T1 model and ensemble models.

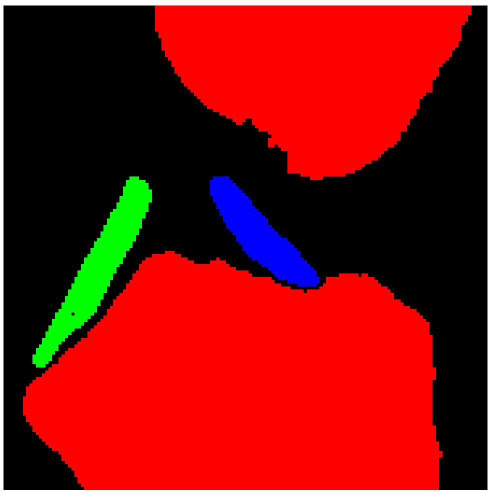
	Accuracy	**Dice**	**Jaccard**
Resolution 1	0.99455 ± 0.00211	0.98911 ± 0.00422	0.97854 ± 0.00812
Resolution 2	0.99489 ± 0.00181	0.98970 ± 0.00356	0.97983 ± 0.00701
Resolution 3	0.99536 ± 0.00144	0.99071 ± 0.00288	0.98165 ± 0.00560
Resolution 4	0.99341 ± 0.00333	0.98682 ± 0.00665	0.97415 ± 0.01284
Resolution 5	0.99454 ± 0.00232	0.98909 ± 0.00463	0.97850 ± 0.00901

Table 6.3: Average scores for the models trained with different input resolutions.

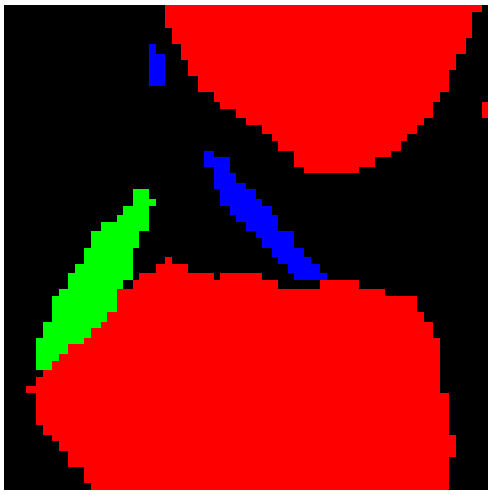
A visual comparison of the weighted ensembles can be seen in figure 6.8. From this example we do see that the higher resolution models have produced a more noisy segmentation mask. The higher resolution segmentation masks contain artefacts like small islands and mislabelling. The lower resolution mask suffers from pixelation and does not appear as smooth as the higher resolution masks. The higher resolution models also seem to be able to capture some of the finer details. For example, how the PCL is detached from the tibia. For the eager reader, section 9.4 in the appendix presents how all the models of the experiment performed on the same example in figure 6.8.

	Accuracy	**Dice**	**Jaccard**
fs 1	0.99051 ± 0.00585	0.98102 ± 0.01171	0.96301 ± 0.02222
pd 1	0.99445 ± 0.00192	0.98890 ± 0.00386	0.97808 ± 0.00754
t1 1	0.99630 ± 0.00049	0.99261 ± 0.00097	0.9853 ± 0.00192
ens 1	0.99574 ± 0.00124	0.99149 ± 0.00249	0.98313 ± 0.00488
wens 1	0.99576 ± 0.00123	0.99152 ± 0.00246	0.98320 ± 0.00483
fs 2	0.99159 ± 0.00498	0.98317 ± 0.00996	0.96709 ± 0.01903
pd 2	0.99430 ± 0.00190	0.98860 ± 0.00380	0.97749 ± 0.00741
t1 2	0.99649 ± 0.00063	0.99298 ± 0.00126	0.98607 ± 0.00248
ens 2	0.99602 ± 0.00136	0.99205 ± 0.00273	0.98424 ± 0.00535
wens 2	0.99604 ± 0.00135	0.99207 ± 0.00271	0.98428 ± 0.00532
fs 3	0.99267 ± 0.00404	0.98533 ± 0.00809	0.97121 ± 0.01565
pd 3	0.99505 ± 0.00181	0.99010 ± 0.00361	0.98042 ± 0.00707
t1 3	0.99657 ± 0.00086	0.99314 ± 0.00173	0.98638 ± 0.00341
ens 3	0.99624 ± 0.00145	0.99249 ± 0.00290	0.98510 ± 0.00570
wens 3	0.99625 ± 0.00145	0.99249 ± 0.00289	0.98512 ± 0.00569
fs 4	0.99094 ± 0.00600	0.98188 ± 0.01200	0.96467 ± 0.02279
pd 4	0.98807 ± 0.00408	0.97613 ± 0.00815	0.95350 ± 0.01558
t1 4	0.99648 ± 0.00093	0.99295 ± 0.00185	0.98601 ± 0.00365
ens 4	0.99577 ± 0.00147	0.99153 ± 0.00295	0.98322 ± 0.00579
wens 4	0.99580 ± 0.00146	0.99160 ± 0.00292	0.98336 ± 0.00573
fs 5	0.99193 ± 0.00415	0.98385 ± 0.00830	0.96834 ± 0.01593
pd 5	0.99151 ± 0.00284	0.98303 ± 0.00568	0.96668 ± 0.01095
t1 5	$\mathbf{0.99672 \pm 0.00074}$	$\mathbf{0.99344 \pm 0.00148}$	$\mathbf{0.98697 \pm 0.00291}$
ens 5	0.99627 ± 0.00109	0.99254 ± 0.00219	0.98520 ± 0.00430
wens 5	0.99629 ± 0.00108	0.99259 ± 0.00216	0.98529 ± 0.00425

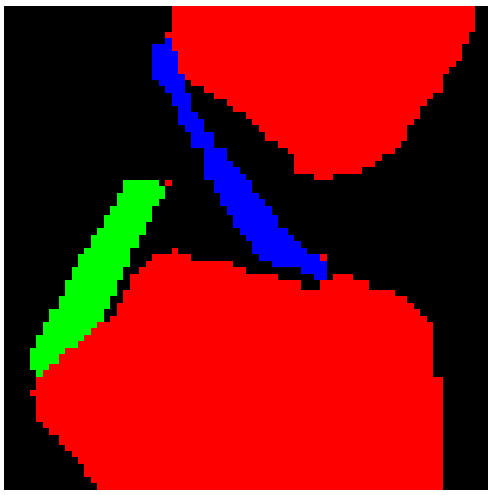
Table 6.4: Resolution experiment all results.



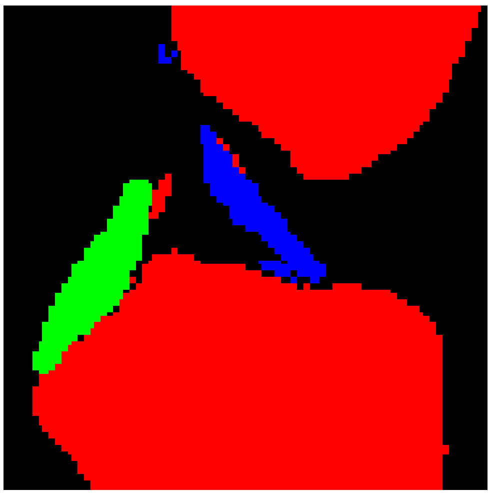
(a) Ground truth



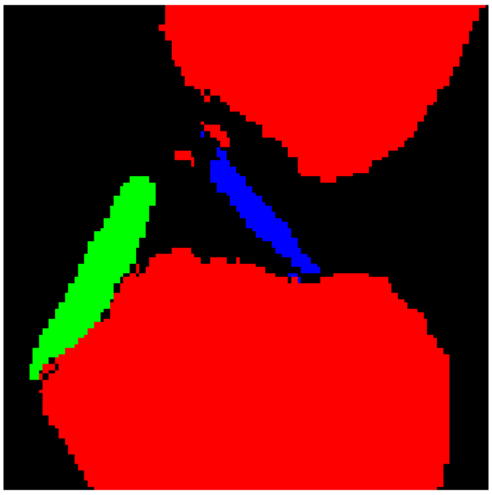
(b) Weighted ensemble 1



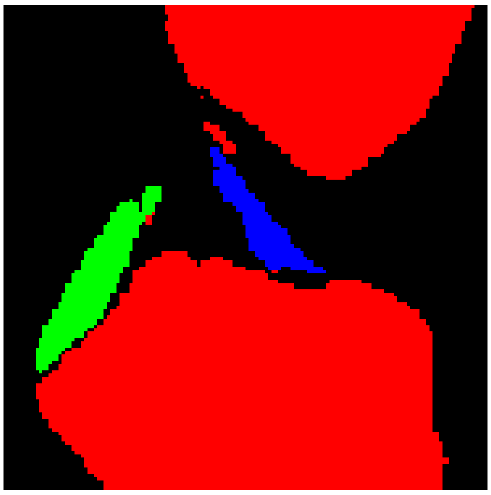
(c) Weighted ensemble 2



(d) Weighted ensemble 3



(e) Weighted ensemble 4



(f) Weighted ensemble 5

Figure 6.8: Zoomed in example showing output mask from the weighted ensemble models trained using different input resolutions.

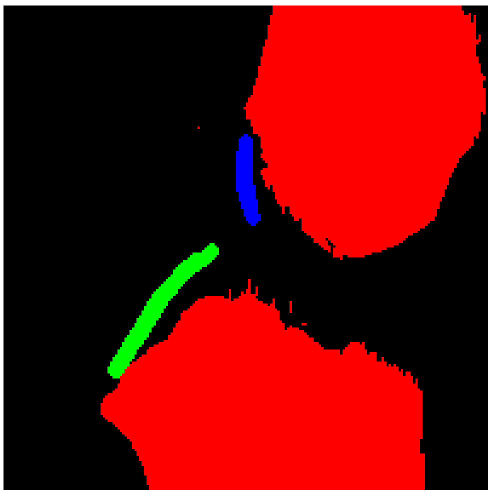
6.5 Fragmented segmentation experiment

The numerical results for fragmented segmentation experiment can be seen in table 6.5. We see that the best model for this experiment scored somewhat lower than the best model of the resolution experiment. Also, we noticed a similar trend to the resolution experiment, where the FS models performed the worst, and the T1 performed the best. However, we do see that the ensemble model performs more similarly to the T1 model for this experiment.

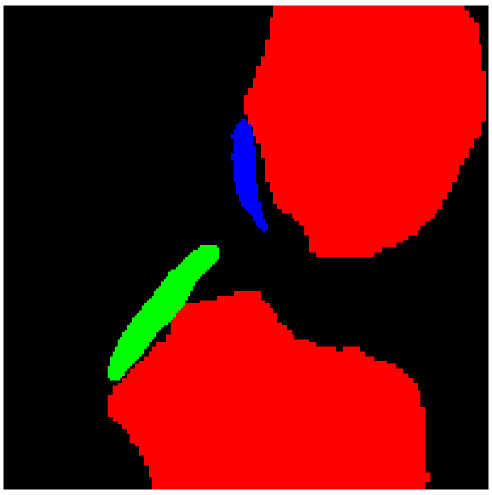
	Accuracy	**Dice**	**Jaccard**
fs	0.99256 ± 0.00412	0.98513 ± 0.00824	0.97082 ± 0.01594
pd	0.99590 ± 0.00183	0.99181 ± 0.00365	0.98377 ± 0.00716
t1	$\mathbf{0.99665 \pm 0.00073}$	$\mathbf{0.99329 \pm 0.00146}$	$\mathbf{0.98668 \pm 0.00288}$
ens	0.99652 ± 0.00134	0.99305 ± 0.00267	0.98621 ± 0.00526

Table 6.5: Accuracy comparison between different fragmented models

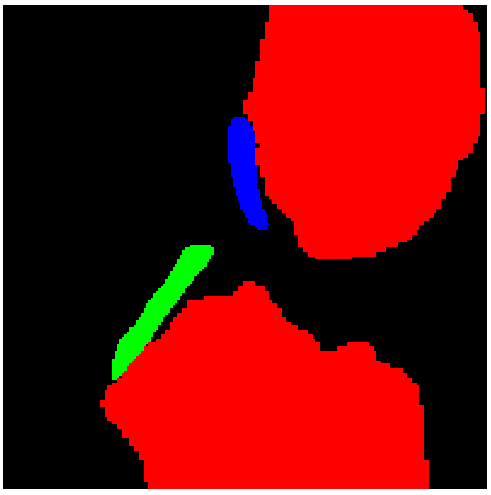
From figure 6.9 we see from a zoomed in example that the fragmented model produces higher resolution masks for the ACL and PCL segment classes compared to the bone mask. The ACL and PCL masks appear less jagged overall than the bone mask. Even though this is another example, we see little to no noise in the mask compared to the masks produced by the models for the resolution experiment. The low noise levels might be a result of the fragmentation of the models, which makes each model more focused on the segmentation of the associated segment class.



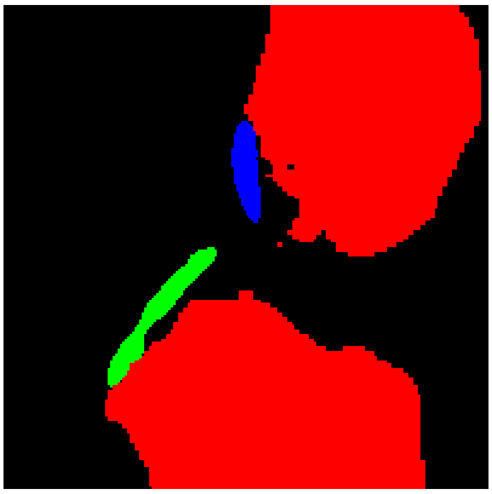
(a) Ground truth



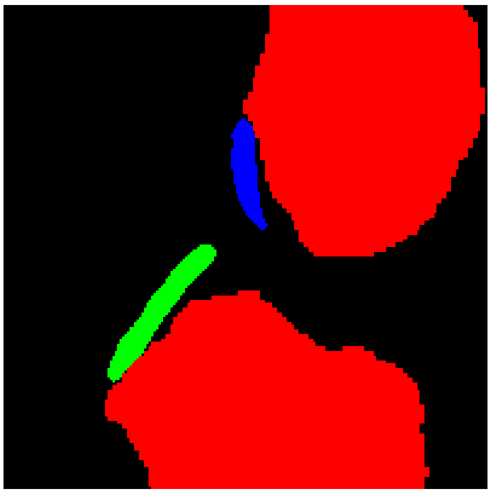
(b) FS Model



(c) PD Model



(d) T1 Model



(e) Ensemble Model

Figure 6.9: Zoomed in output examples from different assemble models.

6.6 Dense conditional random fields

It was found that the score sometimes deteriorated using the same DCRF parameters on various models. Because of this, a parameter search seemed necessary for each model. As a consequence, the DCRF were only tested on two models. The first is the best model from the resolution experiment, the T1 model with an $192 \times 288 \times 288$ input resolution. This model will now be abbreviated T1*.

The second model to test, was the fragmented model, as it produced high-resolution segmentation masks for the ACL and PCL segment classes. But because the resolution experiment typically showed superior results for the models when using higher resolution inputs, the fragmented model was updated. The models segmenting the bones were replaced with new models having $192 \times 288 \times 288$ as the input resolution. Instead of using only the T1 models, the ensembles were used. The contribution of each model was also weighted since the weighted ensemble improved the score slightly. For simplicity, the aforementioned model will go by the name updated model.

The results using the DCRF are presented in table 6.6. We see that the DCRF does not improve the updated models scores by much, but improves the T1* a small amount. We found the best prediction mask of the T1* model achieved a Dice score of 0.99553, and the worst a Dice score of 0.99270. Updating the model gave the best prediction mask a Dice score of 0.99589 and the worst prediction Dice score of 0.98844.

In figure 6.10 we see an example of how the DCRF inference model filters the segmentation map produced by the T1* model. Even though the segmentation map is quite confident about its voxel labels, we can see how refined the confidence map is around the contour of the various shapes. Also, the small island near the patella is filtered away. Furthermore, the noise around the PCL and ACL area for the bone confidence map has been filtered.

	Accuracy	**Dice**	**Jaccard**
Updated	0.99673 ± 0.00128	0.99347 ± 0.00256	0.98703 ± 0.00503
Updated DCRF	0.99675 ± 0.00131	0.99351 ± 0.00262	0.98711 ± 0.00516
T1*	0.99672 ± 0.00074	0.99344 ± 0.00148	0.98697 ± 0.00291
T1* DCRF	0.99723 ± 0.00052	0.99445 ± 0.00103	0.98897 ± 0.00204

Table 6.6: Results from the DCRF experiment.

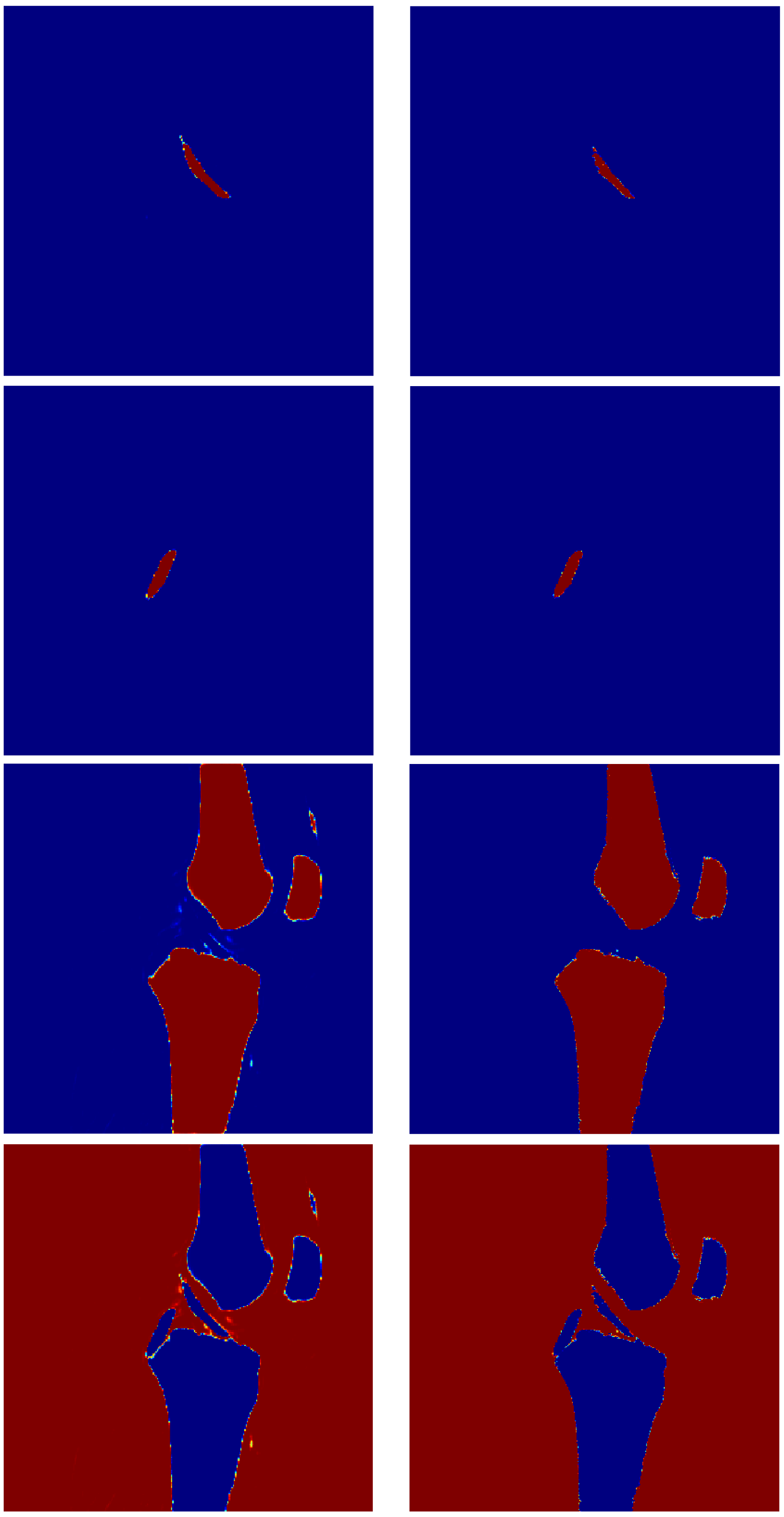


Figure 6.10: Example heatmaps for DCRF. The left column shows the heatmap and the unary energy before the DCRF inference. The second column shows the final segmentation map after the DCRF inference. The first row illustrates the heatmaps for ACL, the second PCL, the third bone, and the fourth background.

6.7 Visual inspection

In this section we will take a look at some of the segmentation masks generated using the models discussed in the previous section to select the best model. It is important to inspect the segmentation masks visually, as the numerical scoring can be deceiving. As the PCL and ACL classes are astronomically small they were also hard to numerically score. A more generalized model, producing a smoother output mask is preferred over a model producing higher scoring noisy masks.

It was found that some of the masks was shifted to the right of the contour of the T1 weighted image, as is illustrated in figure 6.11. This problem only occurred with the updated model, and not the T1* model. On further inspection it was found that some of the PD and FS weighted images were not correctly aligned with the T1 weighted images, which likely made the PD and FS models predict the segmentation mask further to the right than it should be. After removing the two most severely misaligned samples from test dataset, the updated model achieved an average Dice score of 0.99507.

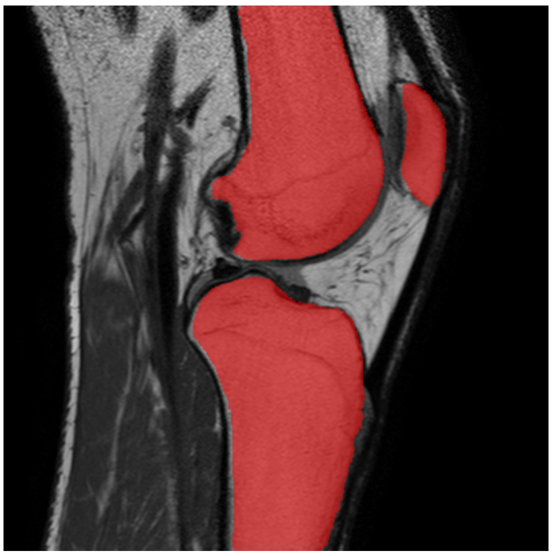
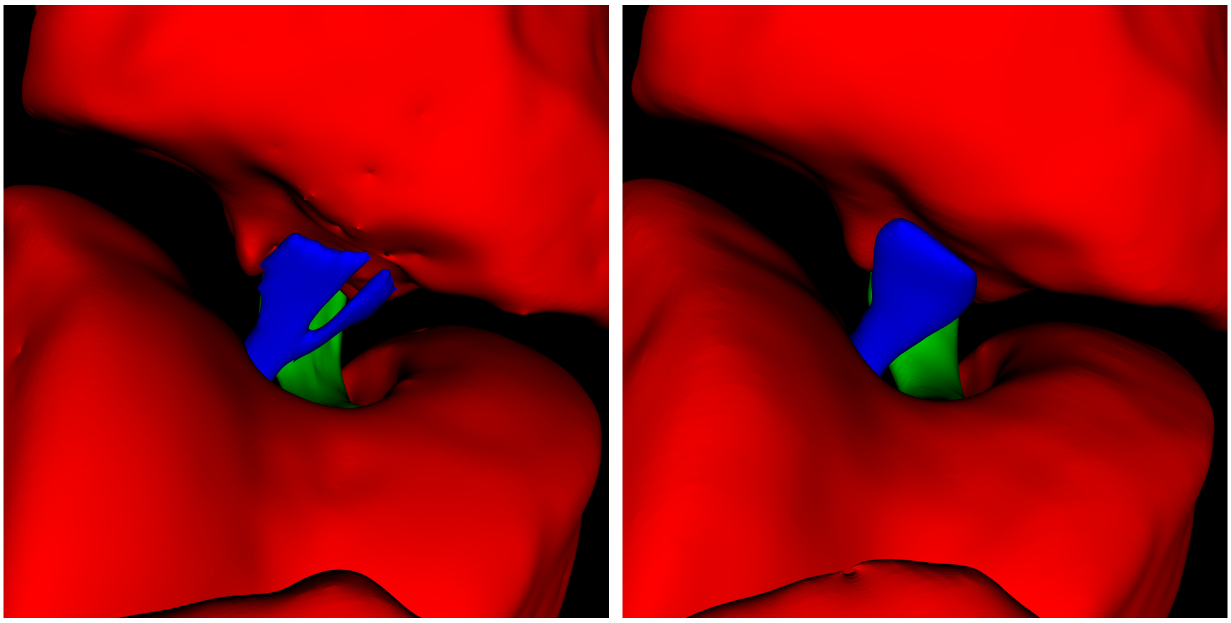


Figure 6.11: Shifted segmentation mask

When investigating the results further, we see that the models seem to generalize more, and are unable to capture some of the fine details of the segmentation masks. One example of this is illustrated in the following figure.



(a) Ground truth mask (b) Fragmented models prediction mask

Figure 6.12: ACL Comparison

This generalization also “fixed” some of the human errors present in the test data. When comparing the ground truth masks to the prediction masks for the final evaluation, it was found some artifacts and human errors in the ground truth masks. The most common types of artifacts in the ground truth images where small inconsistencies and non-smooth surfaces. Two instances of human error where the segmentation mask did not cover the whole upper part of the femur bone of the images. Another human oversight was how some of the ground truth masks were very noisy, having small floating islands of voxels randomly positioned throughout the volume. Figure 6.13 illustrates an example of a ground truth mask containing much noise. On inspection, we found that our models did not reproduce these artifacts, and the segmentation masks produced contained less noise than their ground truth counterparts.

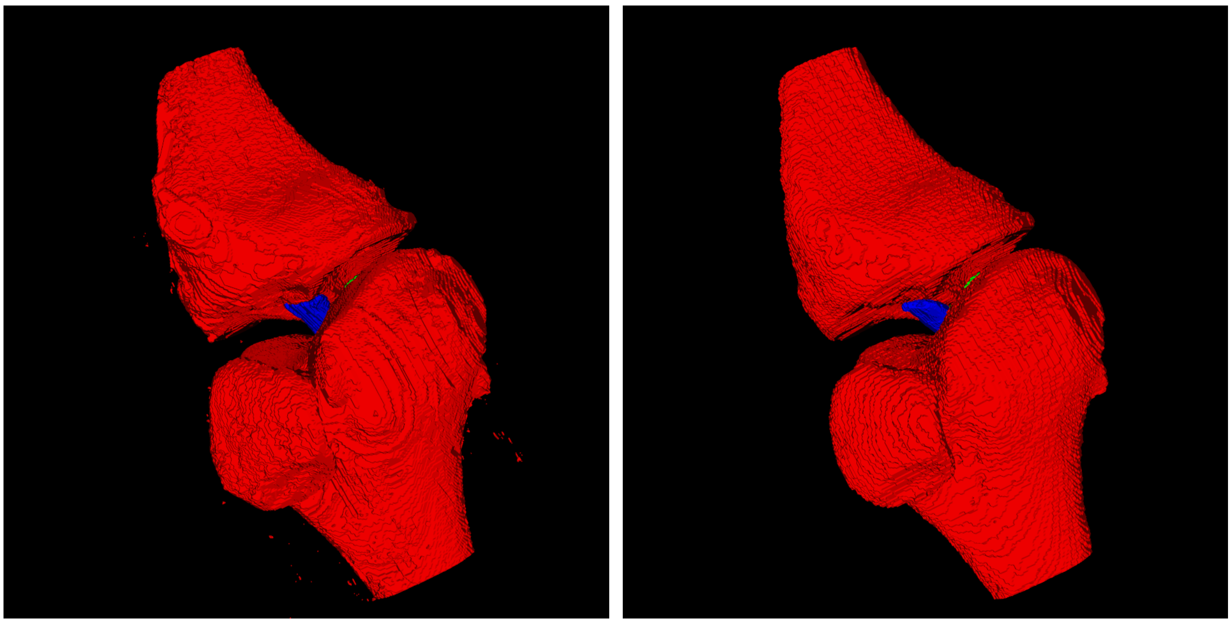

(a) Ground truth mask (b) Fragmented models prediction mask

Figure 6.13: Example of noisy ground truth mask compared to prediction.

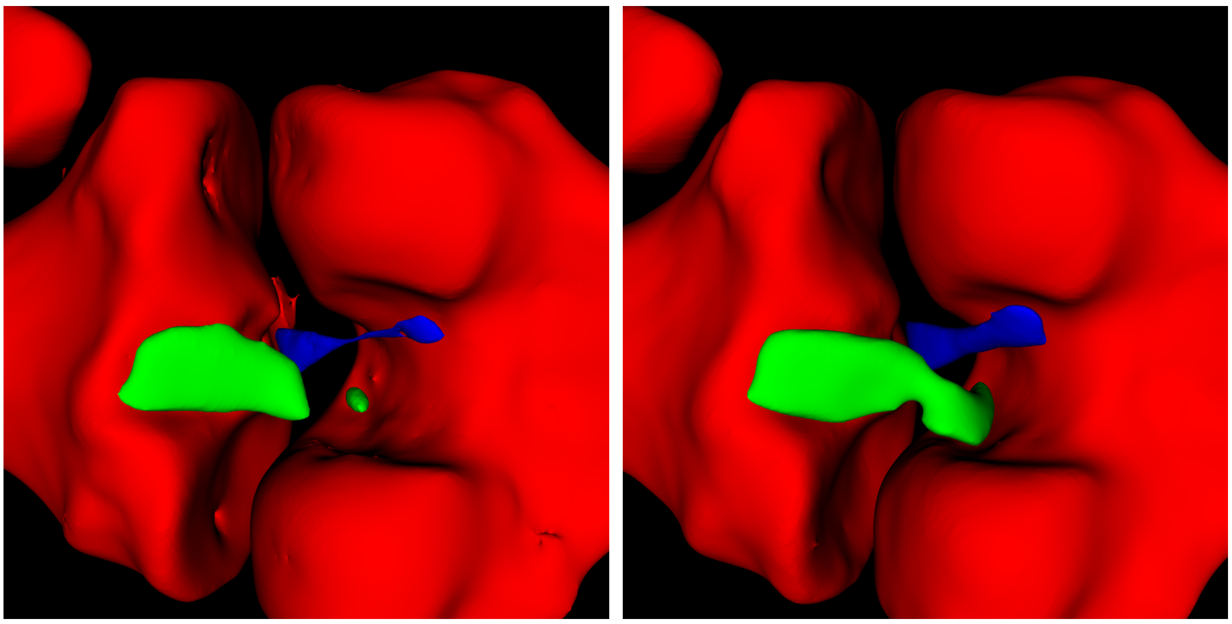

(a) T1* model prediction mask (b) Fragmented model prediction mask

Figure 6.14: Mask comparing

When comparing the updated model to the T1* model, we found that the results of the updated model were producing superior segmentation masks, even though the T1* model scored higher. In figure 6.14we can see an example where the T1* model produced an inferior mask for the PCL and ACL segment classes. And for this example, the fragmented model achieved a Dice score of 0.99075, while the T1* model scored 0.99522. We see that the updated model produces a superior segmentation mask for this specific knee, despite having a lower score.

6.7.1 Anomalies

One anomaly that was only present in one sample was how parts of the lower portion of the fibula were missing. This anomaly occurred in both the updated and the T1* model, and is shown in figure 6.15a and 6.15b. The anomaly was less severe in the segmentation mask produced by the fragmented model. On inspection of the T1, PD and FS weighted images, we saw that the voxels at the lower part of the fibula on this particular sample was a bit darker compared to other samples.

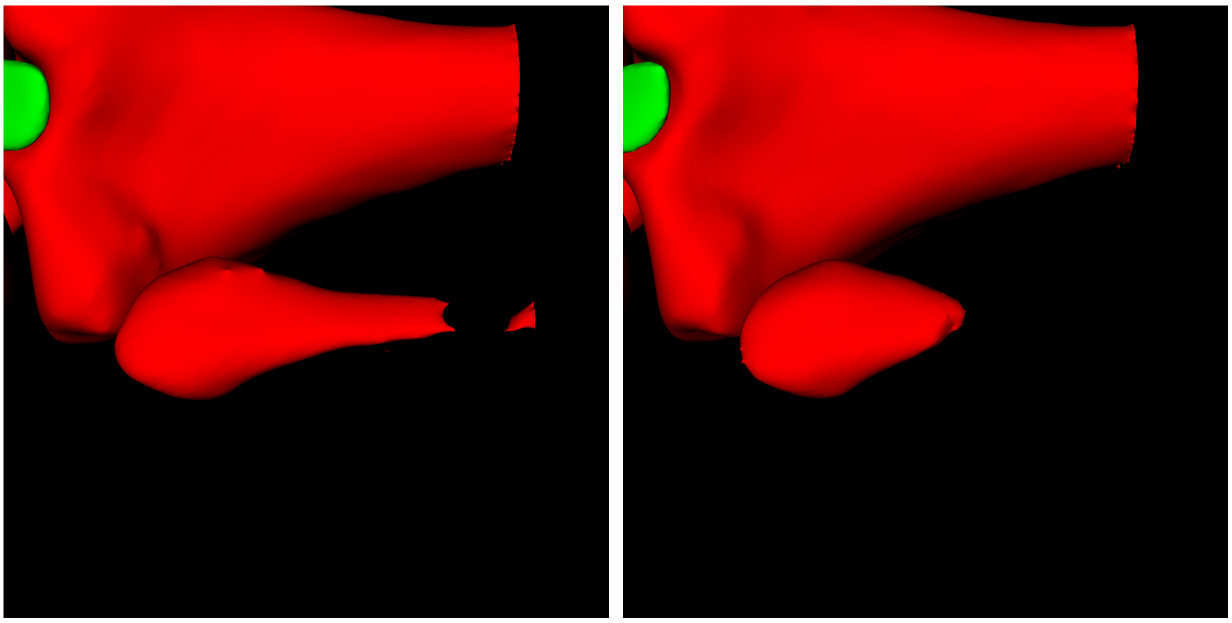

(a) Fragmented model (b) T1* model

Figure 6.15: Fibula anomaly

Another anomaly that was present in one of the testing samples was an extra part of bone, in the proximity of the fibula. This extra bone might be an accessory bone or supernumerary bone. This anatomical anomaly is not common. The anomaly is illustrated in figure 6.16, and can be seen on the left side of the fibula head. None of the trained models segmented the extra bone. This was probably because none of the training samples had any similar anatomical anomalies.

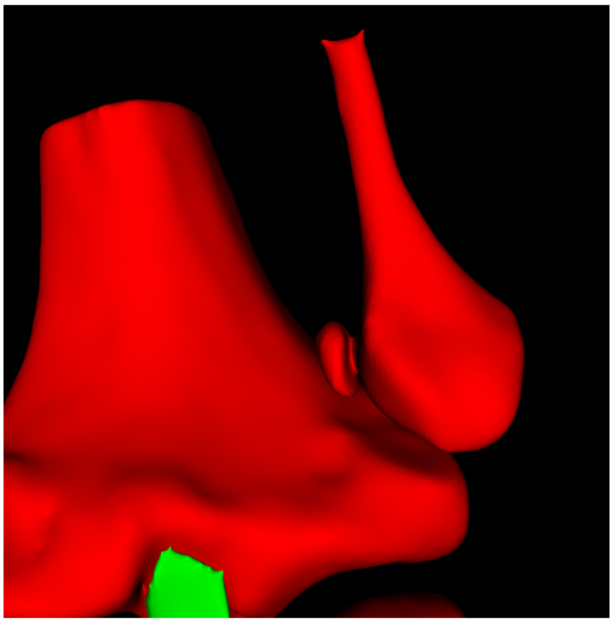
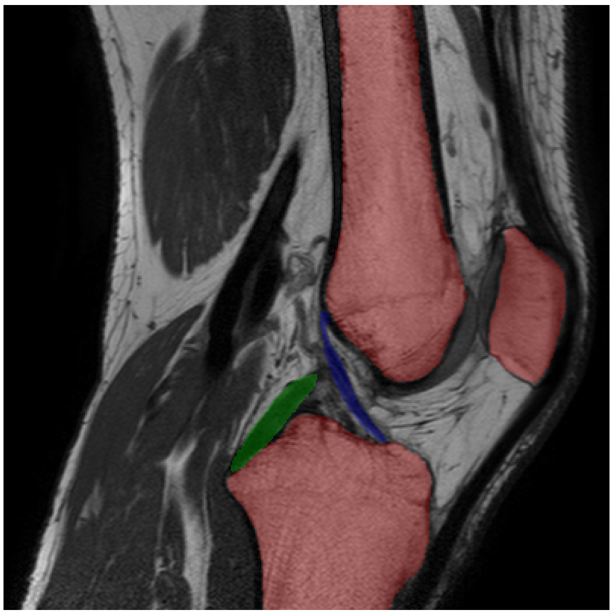
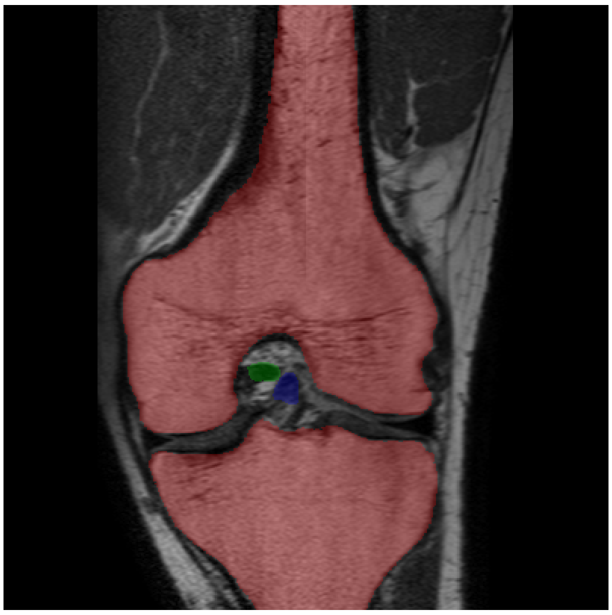


Figure 6.16: Fibula accessory bone anomaly

6.7.2 Final results

The best scoring segmentation mask from the fragmented model is presented in 2D in figure 6.17, and in 3D in figure 6.18. For the eager reader, all the fragmented models predicted segmentation masks are compared to their ground truth counterparts in section 9.5 of the appendix.


(a) Example slice in sagittal plane


(b) Example slice in coronal plane

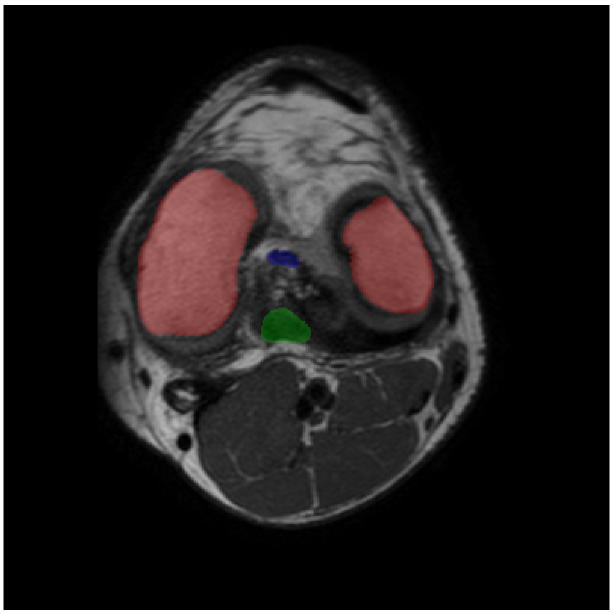
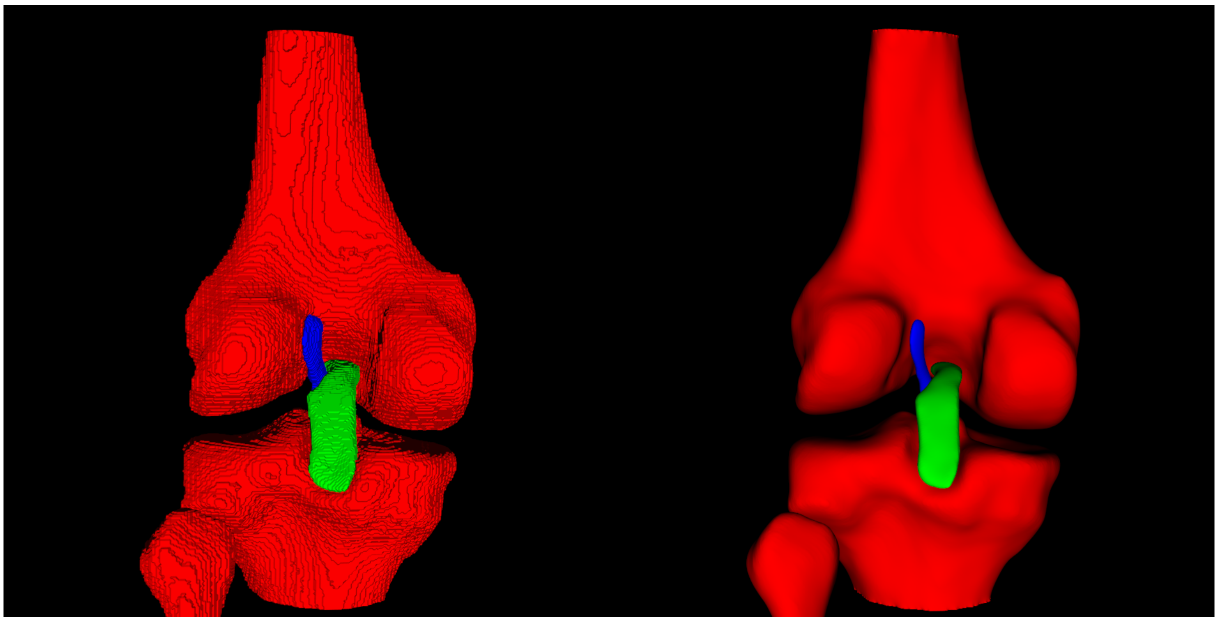

(c) Example slice in axial plane

Figure 6.17: Example slices of best segmented mask.



(a) Segmented mask in 3D (b) Segmented mask in 3D smoothed

Figure 6.18: 3D presentations of the best segmented mask.

Chapter 7: Discussion

This chapter will discuss and evaluate the proposed methods and also assess the results concerning the research questions. In section 7.1, the proposed methods are discussed and evaluated. Furthermore, in section 7.2, we discuss the research questions for this thesis, which were stated in the introduction of the thesis.

7.1 Evaluation

This section will discuss the results and findings of the methods, tools and the experiments conducted.

7.1.1 Segmentation configuration platform

The lightweight MRI segmentation platform developed for this thesis has proven to be useful for the segmentation task of this thesis. The platform was heavily used for experimentation and simplified the workflow significantly. However, the platform was developed specifically around the knee joint segmentation task and has not been tested on any other segmentation tasks. The implemented features and methods are tailored and selected for this specific task. Adapting the platform for other segmentation tasks would require some work, but should not present any significant problems. Given a new segmentation task new features may also be necessary, but due to the flexibility of the platform, it should be possible to extend.

7.1.2 Segmentation evaluation tools

The segmentation evaluation tools were used extensively throughout the final testing stages of the thesis. However, the features of the tool are tailored to the needs for this thesis and might lack features necessary for other tasks. The 3D visualization was quite useful, as it was possible to inspect whole segmentation masks very quickly without having to check the slices individually. The 2D inspection tool was beneficial when comparing how well the segmentation masks fit their real MRI image counterparts. The evaluation tool was also quite valuable, as it was possible to see where in a predicted volume and which slices were the least accurate.

7.1.3 Data augmentation

The data augmentation methods improved the model's ability to generalize. We saw that the model trained on the non-augmented dataset was overfitting by some degree. The segmentation task does not seem to be dependent on a large amount of data. This is probably because the data is very similar, and because the U-Net is a fully convolutional neural network, which is quite robust against overfitting. Given more data, the augmentation methods may be unnecessary.

7.1.4 Loss functions and loss experiment

The results showed that many of the loss functions were incapable of training the model. The reason might be that the loss functions were not able to deal with the heavy class imbalance present in the dataset. Another reason could be that the hyper-parameters used for our networks might work for some loss functions, but not for others.

We found that both the WDL and the WJL combined with the WCCL function performed very similar. The difference between their performance was negligible and was within margin of error. We also found that the WCCL performed similar compared to the combination functions, but required significantly more training epochs to converge.

It was somewhat unexpected that the model using GDL function was not able to get past the local minimum. The only difference from the WDL function proposed are the way it is weighted. The GDL weighting scheme weights the smaller segment classes higher than the proposed WDL function. The difference in weighting could potentially be the reason why the model trained using the WDL was somewhat successful, and the model trained using the GDL was not.

It was found that a combination of weighted categorical cross-entropy and weighted overlap term for the loss function using the appropriate weighting was able to deal with the massive class imbalance of the data. This is probably because of the combination of an overlapping term and a cross-entropy term

7.1.5 Resolution experiment

From the resolution experiment, we found that the 3D U-Net did not require many parameters to achieve sufficient accuracy. This discovery was somewhat unexpected as the MRI images of the knee joints are large and contain a lot of information.

We found that using an input resolution of $128 \times 192 \times 192$ performed best on average. A balance between model parameters and input resolution yields the optimal performance. The reason for this is probably because using too few filter parameters the network is not able to properly learn the underlying features of the images, and by using a lower input resolution prunes too much of the information in the images.

However, we also found that the ensemble models benefited from using higher input resolution. And on inspection, we found that the ensemble models were able to average out a lot of the errors the individual models produced. We also found that the weighted ensemble models did not seem to improve the performance by much. However, it might be possible to further improve the performance of the weighted ensemble by doing a weighting search trying different weightings to find the optimal weightings of the model's contribution to the ensemble.

From the experiment, we also found how the models trained on different weighted MRI images compared. Because of the alignment issue, the results are not entirely accurate. However, it could be the case that the models trained on the T1 images are better overall as they seem to highlight the anatomical structures better.

7.1.6 Fragmented model

It seems like it is beneficial to split up the segmentation task into several tasks for each segment class. We found that the model was better at segmenting the ACL and PCL classes than the other models. This is probably due to the higher number of parameters for the models trained on these classes. By splitting up the segmentation task, we were also able to produce higher resolution segmentation masks.

However, comparing the fragmented model to the other models, there is a trade-off between model accuracy, complexity, training and inference time, and output resolution. Where the fragmentation method is far more complicated as it is necessary to train separate models for each segment class, which increases the training and inference time and also take up more space on the hard-drive as the weights for each model has to be stored. The fragmented models also introduce an extra step of pre-processing as the cropping dimensions for the different classes had to be found.

7.1.7 Dense conditional random fields

It seems like the 3D DCRF was somewhat unnecessary for the fragmented model. The DCRF process was only able to improve the accuracy of the fragmented model by an insignificant amount. This lack of improvement was probably because of the ensemble averaging had already dealt with filtering out the noisy parts of the segmentation map, and the segmentation map's appearance.

When using the DCRF process over the segmentation maps produced by the T1* model, the Dice score improved to some extent. The DCRF inference process seem more beneficial for the individual models than the ensemble models.

The parameter search that was performed was short and limited to a small range for each parameter. Performing a broader search could potentially find better-suited parameters, and consequently, improve the performance of the DCRF inference processes further.

As previously discussed the inference process of the DCRF models was also quite slow. However, speed is not critical for the segmentation process. It should not be too much of an issue when used in a real-world application. Another problem with the DCRF inference process was memory usage. The DCRF process required a lot of system memory when inferring over the volume using the original resolution.

7.1.8 Final Results

It was recognized a bit too late that some of the MRI images were not properly aligned. It was assumed that the data provided was correctly aligned for all the images. And because the ground truth segmentation masks are painted using the T1 weighted images as background templates, the results are a bit skewed and biased towards models trained on the T1 data. However, there was insufficient time to do an extensive investigation of the severity of the issue, and if the issue affected the training sets as well. Given more time, it would be interesting to see if the results would be any different with a new realigned dataset. Nonetheless, we could see from the results that the proposed methods produced good segmentation masks.

We saw that the ensemble averaging did not improve the numerical result by much, but upon inspection, it was observed that the masks produced by the ensemble model contained less noise. However, the weighting did not improve the score appreciably. A search could be implemented to learn the appropriate weights for the different models in the ensemble.

Because of the alignment problem, it could be argued that it would be better to only ensemble models trained on the T1 weighted images. However, by ensembling models trained on different weighted MRI images, we can utilize different representations of the anatomical structures for a less biased segmentation map.

On inspection, the masks looked reasonable. The segmentation masks do not have to be perfect, as they can be modified and fixed at a later stage. The machine learning algorithm gives a good estimation. It was also noted during meetings and consultation that the algorithm, in some cases, gave a better segmentation in some areas than the ground truth.

7.2 Goals

The overall research goal of this thesis was to propose and develop methods and tools for segmentation of MRI images of the knee joint anatomy using deep learning techniques. It was desired that the methods for use for the segmentation would yield relatively good accuracy for the segmentation task.

7.2.1 Segmentation accuracy

The proposed segmentation methods achieved an overall Dice score of 0.995 on the segmentation task for the best model. As previously discussed, semantic segmentation can be challenging to score numerically. On inspection, we saw that the segmentation masks produced by our models were quite good. We also found that the segmentation masks produced by the proposed methods were better in some regards.

However, it is not critical that the models produce perfect segmentation masks, as errors are possible to be controlled for and fixed at a later stage. Nevertheless, given new and more data with the issues addressed, we believe the proposed segmentation methods will produce results with even better accuracy. Finally, some of the segmentation masks were evaluated by an expert at from Sunnmøre MR-Klinikk their feedback was positive.

7.2.2 Adaptability

The methods explored in this thesis have been proven to work for segmenting three different segment classes of MRI images of the knee joint anatomy. The methods were unfortunately not tested on any other segmentation task, but given how similar MRI images of different joint structures are, it should be possible to apply the same methods for similar types of data. For example, MRI images of the shoulder joint are probably solvable given the methods proposed.

7.2.3 Real world application

The segmentation comparison tool seems to be somewhat ready for real-world applications, as it has been successfully compiled and tested on five different Windows computers. However, the tool did have some scaling issues for some of the GUI components when tested on laptops. The tool has also not been extensively tested on other data, but the tool was tested on a segmentation example of the liver and was capable of loading it and displaying the data both in 2D and 3D.

For the segmentation platform, the configuration file structures might need some documentation if others are going to use it, as parts of it might not be beginner-friendly. The platform also might need some changes as some of the features were hastily implemented, and are not very intuitive to use.

Chapter 8: Conclusion

In recent years semantic segmentation methods using deep learning methods have been developed for segmentation tasks for different biomedical images. This thesis has further explored how to utilize some of these new methods in regards to MRI images of the anatomical structures of the knee joint. Specifically fully convolutional 3D U-Net and 3D DCRF have been used to produce dense segmentation masks for the MRI images.

Furthermore, several different experiments have been conducted, giving insight and a better understanding of how to further develop and implement methods to use for future segmentation problems. A platform for MRI segmentation has been developed, which allows for designing different model pipelines using configuration file structures. A segmentation tool has been developed for evaluating MRI segmentation masks visually in 2D and 3D.

Some of the preliminary work for the segmentation part of the collaborative project between NTNU, Sunnmøre MR-Klinikk and Ålesund Hospital has been carried out, and a flexible platform for MRI segmentation has been established. The work of this thesis has been a learning experience, and has given insight on where to proceed next for the project, and also what needs to be improved on in the future.

8.1 Contributions

To our knowledge, the proposed combination of loss functions has not been covered in any of the literature. For this problem, the combination of the proposed loss functions seemed to work better than the other functions from literature which they were tested against, but we cannot assume that the proposed loss functions will work better for other segmentation tasks.

8.2 Future work

This task has revealed the potential for a segmentation platform for the collaboration project. However, there is still room for improvements. A list has been made to address aspects of this thesis that can be further improved and explored and can hopefully inspire future work and possibly new ideas for a future thesis. The list of these possible directions for further improvement and development are given below.

- **Test the proposed segmentation methods on new segment classes and new data.** As of the writing of this thesis the segmented classes of the data was limited to bone, the PCL, and the ACL. However, the plan for the collaboration project is to segment the different anatomical classes for the whole knee. The dataset was also tiny, meaning further improvements can be achieved given more data to train the models.

  The collaboration project will also be extended to other parts of the human body, including the shoulders. It would be interesting to see how well the proposed methods in this thesis would perform on new data covering different anatomical parts of the body.

- **Improve the segmentation method by finding the better parameters** A lot of the choices for the parameters of the proposed method were based on intuition because of time constraints. The proposed method can probably be further improved by optimizing the parameters using parameter search methods.

- **Scale the proposed methods on better or future hardware.** Even though the proposed methods are performed on quite good hardware, there is still room for improvement, as demonstrated by the fact that the proposed methods were in some cases limited by GPU memory. To further improve the segmentation mask, we suggest increasing the resolution and the number of network parameters to some extent.

- **Implement and test other CNN architectures.** In this thesis, the U-Net was the focused CNN architecture. However, it could be interesting to see how other CNN architectures would handle this problem in combination the methods proposed in this thesis.

Bibliography

[1] Edward H Adelson and James R Bergen. *The Plenoptic Function and the Elements of Early Vision*. Tech. rep. 1991, pp. 3–20. URL: http://persci.mit.edu/pub_pdfs/elements91.pdf.

[2] Erik Andersson and Robin Berglund. *Evaluation of Data Augmentation of MR Images for Deep Learning*. Tech. rep. URL: http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=8952747&fileOId=8952748.

[3] *Autoencoder_structure.png (677×506)*. URL: https://upload.wikimedia.org/wikipedia/commons/2/28/Autoencoder_structure.png.

[4] Fangliang Bai, Manuel J. Marques, and Stuart J. Gibson. "Cystoid macular edema segmentation of Optical Coherence Tomography images using fully convolutional neural networks and fully connected CRFs". In: (Sept. 2017). URL: http://arxiv.org/abs/1709.05324.

[5] Lucas Beyer. *PyDenseCRF*. https://github.com/lucasb-eyer/pydensecrf. 2015.

[6] Ashwin Bhandare et al. *Applications of Convolutional Neural Networks*. Tech. rep. URL: www.ijcsit.com.

[7] Zuzana Bílková. *MASTER THESIS Segmentation of Microscopic Images Using Level Set Methods*. Tech. rep. URL: http://www.utia.cas.cz/files/Diplomov%C3%A1%20pr%C3%A1ce%2C%20Zuzana%20B%C3%ADlkov%C3%A1.pdf.

[8] Leo Breiman. "Bagging predictors". In: *Machine Learning* 24.2 (Aug. 1996), pp. 123–140. ISSN: 0885-6125. DOI: 10.1007/BF00058655. URL: http://link.springer.com/10.1007/BF00058655.

[9] Matthew Brett et al. *NiBabel Access a cacophony of neuro-imaging file formats.* https://nipy.org/nibabel/. 2019.

[10] Tom Brosch et al. "LNCS 9351 - Deep Convolutional Encoder Networks for Multiple Sclerosis Lesion Segmentation". In: (2015). DOI: 10.1007/978-3-319-24574-4{_}1. URL: http://www.rogertam.ca/Brosch_MICCAI_2015.pdf.

[11] Jerrold T. Bushberg et al. *The essential physics of medical imaging*, p. 1030. ISBN: 9780781780575. URL: https://radiopaedia.org/articles/pd-weighted-spin-echo-images?lang=us.

[12] Luke Campagnola. *PyQtGraph: Scientific Graphics and GUI Library for Python.* http://www.pyqtgraph.org. 2019.

[13] *Cardiac MRI | Stanford Health Care.* URL: https://stanfordhealthcare.org/medical-tests/m/mri/types/cardiac-mri.html.

[14] François Chollet et al. *Keras.* https://keras.io. 2015.

[15] Patrick Ferdinand Christ et al. "Automatic Liver and Lesion Segmentation in CT Using Cascaded Fully Convolutional Neural Networks and 3D Conditional Random Fields". In: (Oct. 2016). DOI: 10.1007/978-3-319-46723-8{_}48. URL: http://arxiv.org/abs/1610.02177%20http://dx.doi.org/10.1007/978-3-319-46723-8_48.

[16] Özgün Çiçek et al. "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation". In: (June 2016). URL: http://arxiv.org/abs/1606.06650.

[17] Marc Claesen and Bart De Moor. "Hyperparameter Search in Machine Learning". In: (Feb. 2015). URL: http://arxiv.org/abs/1502.02127.

[18] Dawn C. Collier et al. "Assessment of consistency in contouring of normal-tissue anatomic structures". In: *Journal of Applied Clinical Medical Physics* 4.1 (Jan. 2003), p. 17. ISSN: 15269914. DOI: 10.1120/1.1521271. URL: http://www.ncbi.nlm.nih.gov/pubmed/12540815%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5724429%20http://www.jacmp.org/archive.php?doi=10.1120%2F1.1521271.

[19] Riverbank Computing. *PyQt.* 2019. URL: https://www.riverbankcomputing.com/software/pyqt/intro.

[20] *Conv_layer.png (634×426)*. URL: https://upload.wikimedia.org/wikipedia/commons/6/68/Conv_layer.png.

[21] T.F. Cootes, G.J. Edwards, and C.J. Taylor. "Active appearance models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.6 (June 2001), pp. 681–685. ISSN: 01628828. DOI: 10.1109/34.927467. URL: http://ieeexplore.ieee.org/document/927467/.

[22] Douglas Crockford. *JavaScript Object Notation (JSON)*. https://json.org. 1999.

[23] W.R. Crum, O. Camara, and D.L.G. Hill. "Generalized Overlap Measures for Evaluation and Validation in Medical Image Analysis". In: *IEEE Transactions on Medical Imaging* 25.11 (Nov. 2006), pp. 1451–1461. ISSN: 0278-0062. DOI: 10.1109/TMI.2006.880587. URL: http://www.ncbi.nlm.nih.gov/pubmed/17117774%20http://ieeexplore.ieee.org/document/1717643/.

[24] Jifeng Dai, Kaiming He, and Jian Sun. "Instance-aware Semantic Segmentation via Multi-task Network Cascades". In: (Dec. 2015). URL: http://arxiv.org/abs/1512.04412.

[25] Emmanuelle M. Delfaut et al. "Fat Suppression in MR Imaging: Techniques and Pitfalls". In: *RadioGraphics* 19.2 (Mar. 1999), pp. 373–382. ISSN: 0271-5333. DOI: 10.1148/radiographics.19.2.g99mr03373. URL: http://pubs.rsna.org/doi/10.1148/radiographics.19.2.g99mr03373.

[26] Lee R. Dice. "Measures of the Amount of Ecologic Association Between Species". In: *Ecology* 26.3 (July 1945), pp. 297–302. ISSN: 00129658. DOI: 10.2307/1932409. URL: http://doi.wiley.com/10.2307/1932409.

[27] John Duchi et al. *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*. Tech. rep. URL: https://stanford.edu/~jduchi/projects/DuchiHaSi10_colt.pdf.

[28] Vincent Dumoulin, Francesco Visin, and George E P Box. *A guide to convolution arithmetic for deep learning*. Tech. rep. 2018. URL: http://ethanschoonover.com/solarized.

[29] Chenyou Fan. *Survey of Convolutional Neural Network*. Tech. rep. URL: http://homes.sice.indiana.edu/fan6/docs/cnn_survey.pdf.

[30] Fred Flandry and Gabriel Hommel. *Normal Anatomy and Biomechanics of the Knee*. Tech. rep. 2011. URL: www.sportsmedarthro.com.

[31] *Functional MRI (fMRI) | Stanford Health Care*. URL: https://stanfordhealthcare.org/medical-tests/m/mri/types/fmri.html.

[32] Josiah Willard Gibbs. *Elementary Principles in Statistical Mechanics*. New York: Charles Scribner's Sons, 1902.

[33] Eli Gibson et al. "NiftyNet: a deep-learning platform for medical imaging". In: 2018. DOI: https://doi.org/10.1016/j.cmpb.2018.01.025. URL: https://www.sciencedirect.com/science/article/pii/S0169260717311823.

[34] Eli Gibson et al. "NiftyNet: a deep-learning platform for medical imaging." In: *Computer methods and programs in biomedicine* 158 (May 2018), pp. 113–122. ISSN: 1872-7565. DOI: 10.1016/j.cmpb.2018.01.025. URL: http://www.ncbi.nlm.nih.gov/pubmed/29544777%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5869052.

[35] Mohammad-Hossein Golbon-Haghighi et al. "Pattern Synthesis for the Cylindrical Polarimetric Phased Array Radar (CPPAR)". In: *Progress In Electromagnetics Research* 66 (2018), pp. 87–98. ISSN: 1937-8726. DOI: 10.2528/PIERM18011016. URL: http://www.jpier.org/PIERM/pier.php?paper=18011016.

[36] T.F. Cootes Graham et al. "Active shape models - their training and application". In: *Computer Vision and Image Understanding* 61 (1995), pp. 38–59. URL: https://en.wikipedia.org/wiki/Active_shape_model#cite_note-Cootes-1.

[37] Umut Güçlü et al. "End-to-end semantic face segmentation with conditional random fields as convolutional, recurrent and adversarial networks". In: (Mar. 2017). URL: http://arxiv.org/abs/1703.03305.

[38] Mohammad Havaei et al. *Brain Tumor Segmentation with Deep Neural Networks $*. Tech. rep. URL: https://arxiv.org/pdf/1505.03540.pdf.

[39] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: (Dec. 2015). URL: http://arxiv.org/abs/1512.03385.

[40] Tobias Heimann and Hans-Peter Meinzer. "Statistical shape models for 3D medical image segmentation: A review". In: *Medical Image Analysis* 13 (2009), pp. 543–563. DOI: 10.1016/j.media.2009.05.004. URL: https://ac.els-cdn.com/S1361841509000425/1-s2.0-S1361841509000425-main.pdf?_tid=4a02d90e-25c3-4f42-a5ee-fb9be5684db4&acdnat=1549291046_8a6a386f89a46320d08d20bc846ad1a4.

[41] Geoffrey Hinton, Ni@sh Srivastava, and Kevin Swersky. *Neural Networks for Machine Learning Lecture 6a Overview of mini--batch gradient descent*. Tech. rep. URL: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[42] Elad Hoffer, Itay Hubara, and Daniel Soudry. *Train longer, generalize better: closing the generalization gap in large batch training of neural networks*. Tech. rep. URL: https://arxiv.org/pdf/1705.08741.pdf.

[43] D. H. Hubel and T. N. Wiesel. "Receptive fields and functional architecture of monkey striate cortex". In: *The Journal of Physiology* 195.1 (Mar. 1968), pp. 215–243. ISSN: 00223751. DOI: 10.1113/jphysiol.1968.sp008455. URL: http://www.ncbi.nlm.nih.gov/pubmed/4966457%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1557912%20http://doi.wiley.com/10.1113/jphysiol.1968.sp008455.

[44] *Human_ anatomy_planes.jpg (2526×3696)*. URL: https://upload.wikimedia.org/wikipedia/commons/7/79/Human_anatomy_planes.jpg.

[45] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing In Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[46] Nabil Ibtehaz and M Sohel Rahman. *MultiResUNet : Rethinking the U-Net Architecture for Multimodal Biomedical Image Segmentation*. Tech. rep. 2019. URL: https://arxiv.org/pdf/1902.04049.pdf.

[47] *Image Segmentation*. URL: https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic3.htm.

[48] *Image Segmentation - MATLAB &amp; Simulink*. URL: https://se.mathworks.com/discovery/image-segmentation.html.

[49] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: (Feb. 2015). URL: http://arxiv.org/abs/1502.03167.

[50] Paul Jaccard. "Étude comparative de la distribution florale dans une portion des Alpes et des Jura". In: *Bulletin de la Société vaudoise des sciences naturelles* 37 (1901), pp. 547–579.

[51] Paul Jaccard. "THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE.1". In: *New Phytologist* 11.2 (Feb. 1912), pp. 37–50. ISSN: 0028-646X. DOI: 10.1111/j.1469-8137.1912.tb05611.x. URL: http://doi.wiley.com/10.1111/j.1469-8137.1912.tb05611.x.

[52] Saumya Jetley et al. "Learn To Pay Attention". In: (Apr. 2018). URL: http://arxiv.org/abs/1804.02391.

[53] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001. URL: http://www.scipy.org/.

[54] Konstantinos Kamnitsas et al. "Efficient Multi-Scale 3D CNN with Fully Connected CRF for Accurate Brain Lesion Segmentation". In: (Mar. 2016). DOI: 10.1016/j.media.2016.10.004. URL: http://arxiv.org/abs/1603.05959%20http://dx.doi.org/10.1016/j.media.2016.10.004.

[55] Dilpreet Kaur and Yadwinder Kaur. *International Journal of Computer Science and Mobile Computing Various Image Segmentation Techniques: A Review*. Tech. rep. 5. 2014, pp. 809–814. URL: www.ijcsmc.com.

[56] Çağrı Kaymak and Ayşegül Uçar. *A Brief Survey and an Application of Semantic Image Segmentation for Autonomous Driving*. Tech. rep. URL: https://arxiv.org/ftp/arxiv/papers/1808/1808.08413.pdf.

[57] Nitish Shirish Keskar et al. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima". In: (Sept. 2016). URL: http://arxiv.org/abs/1609.04836.

[58] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: (Dec. 2014). URL: https://arxiv.org/abs/1412.6980.

[59] *Knee &amp; Leg - Atlas of Anatomy*. URL: https://doctorlib.info/medical/anatomy/27.html.

[60] Philipp Krähenb, Krähenb¨ Krähenbühl, and Vladlen Koltun. *Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials*. Tech. rep. URL: https://arxiv.org/pdf/1210.5644.pdf.

[61] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. Tech. rep. URL: http://code.google.com/p/cuda-convnet/.

[62] John Lafferty et al. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data Part of the Numerical Analysis and Scientific Computing Commons Recommended Citation &quot;Conditional Random Fields: Probabilistic Models for Segmenting and Labelin*. Tech. rep. 2001, pp. 282–289. URL: http://repository.upenn.edu/cis_papersPublisherURL:http://portal.acm.org/citation.cfm?id=655813PublisherURL:http://portal.acm.org/citation.cfm?id=655813ThisconferencepaperisavailableatScholarlyCommons:http://repository.upenn.edu/cis_papers/159.

[63] L. D. (Lev Davidovich) Landau, E. M. (Evgenii Mikhaiilovich) Liftshits, and L. P. (Lev Petrovich) Pitaevskii. *Statistical physics*. 3rd ed. Course of Theoretical Physics. Oxford: Butterworth-Heinemann, c1980, 1996. ISBN: 0750633727.

[64] Michele Larobina and Loredana Murino. "Medical image file formats." In: *Journal of digital imaging* 27.2 (Apr. 2014), pp. 200–6. ISSN: 1618-727X. DOI: 10.1007/s10278-013-9657-9. URL: http://www.ncbi.nlm.nih.gov/pubmed/24338090%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3948928.

[65] Chunming Li et al. "A Level Set Method for Image Segmentation in the Presence of Intensity Inhomogeneities With Application to MRI". In: *IEEE TRANSACTIONS ON IMAGE PROCESSING* 20.7 (2011). DOI: 10.1109/TIP.2011.2146190. URL: http://ieeexplore.ieee.org..

[66] Wenqi Li et al. *On the Compactness, Efficiency, and Representation of 3D Convolutional Networks: Brain Parcellation as a Pretext Task*. Tech. rep. URL: https://arxiv.org/pdf/1707.01992.pdf.

[67] Cheng-Yuan Liou, Jau-Chi Huang, and Wen-Chie Yang. "Modeling word perception using the Elman network". In: *Neurocomputing* 71.16-18 (Oct. 2008), pp. 3150–3157. ISSN: 0925-2312. DOI: 10.1016/J.NEUCOM.2008.04.030. URL: https://www.sciencedirect.com/science/article/pii/S0925231208002865?via%3Dihub.

[68] Cheng-Yuan Liou et al. "Autoencoder for words". In: *Neurocomputing* 139 (Sept. 2014), pp. 84–96. ISSN: 0925-2312. DOI: 10.1016/J.NEUCOM.2013.09.055. URL: https://www.sciencedirect.com/science/article/pii/S0925231214003658?via%3Dihub.

[69] Jonathan Long, Evan Shelhamer, and Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation.* Tech. rep. URL: https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf.

[70] William E. Lorensen and Harvey E. Cline. "Marching cubes: A high resolution 3D surface construction algorithm". In: *ACM SIGGRAPH Computer Graphics* 21.4 (Aug. 1987), pp. 163–169. ISSN: 00978930. DOI: 10.1145/37402.37422. URL: http://portal.acm.org/citation.cfm?doid=37402.37422.

[71] JGraph Ltd. *draw.io.* 2019. URL: https://www.draw.io.

[72] S M Masudur et al. *Shape-aware Deep Convolutional Neural Network for Vertebrae Segmentation.* Tech. rep. URL: http://gregslabaugh.net/publications/ArifMSKI-MICCAI2017.pdf.

[73] Donald W. McRobbie. *MRI from picture to proton.* Cambridge, UK; New York: Cambridge University Press, 2007, p. 394. ISBN: 9780521683845.

[74] Bjoern H Menze et al. "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)." In: *IEEE transactions on medical imaging* 34.10 (Oct. 2015), pp. 1993–2024. ISSN: 1558-254X. DOI: 10.1109/TMI.2014.2377694. URL: http://www.ncbi.nlm.nih.gov/pubmed/25494501%0Ahttp://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4833122.

[75] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. "V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation". In: (June 2016). URL: http://arxiv.org/abs/1606.04797.

[76] Donald G. Mitchell and Mark Cohen. *MRI principles.* Saunders, 2004, p. 400. ISBN: 0721600247.

[77] Melanie (Computer scientist) Mitchell and Melanie. *An introduction to genetic algorithms*. MIT Press, 1996, p. 205. ISBN: 0262133164. URL: https://dl.acm.org/citation.cfm?id=230231.

[78] *MRI scan - NHS*. URL: https://www.nhs.uk/conditions/mri-scan/.

[79] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[80] *Object Detection - MATLAB &amp; Simulink*. URL: https://se.mathworks.com/discovery/object-detection.html.

[81] *Object Recognition - MATLAB &amp; Simulink*. 2017. URL: https://www.mathworks.com/discovery/object-recognition.html..html.

[82] Ozan Oktay et al. "Attention U-Net: Learning Where to Look for the Pancreas". In: (Apr. 2018). URL: http://arxiv.org/abs/1804.03999.

[83] Ingy döt Net Oren Ben-Kiki Clark Evans. *YAML: YAML Ain't Markup Language*. https://yaml.org. 2001.

[84] Stanley Osher and James A Sethian. "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations". In: *Journal of Computational Physics* 79.1 (Nov. 1988), pp. 12–49. ISSN: 00219991. DOI: 10.1016/0021-9991(88)90002-2. URL: https://linkinghub.elsevier.com/retrieve/pii/0021999188900022.

[85] *overfitting | Definition of overfitting in English by Lexico Dictionaries*. URL: https://www.lexico.com/en/definition/overfitting.

[86] *radiologi – Store medisinske leksikon*. URL: https://sml.snl.no/radiologi.

[87] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: (May 2015). URL: http://arxiv.org/abs/1505.04597.

[88] Holger R. Roth et al. "Deep learning and its application to medical image segmentation". In: (Mar. 2018). DOI: 10.11409/mit.36.63. URL: http://arxiv.org/abs/1803.08691%20http://dx.doi.org/10.11409/mit.36.63.

[89] Sebastian Ruder. *An overview of gradient descent optimization algorithms **. Tech. rep. URL: http://caffe.berkeleyvision.org/tutorial/solver.html.

[90] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. ISSN: 00280836. DOI: 10.1038/323533a0. URL: http://www.nature.com/articles/323533a0.

[91] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: (Sept. 2014). URL: http://arxiv.org/abs/1409.0575.

[92] Stuart Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach (3rd Edition)*. 3rd. ISBN: 9780136042594. DOI: 10.1017/S0269888900007724. arXiv: 9809069v1 [gr-qc].

[93] Will. Schroeder et al. *The visualization toolkit : an object-oriented approach to 3D graphics*. Kitware, 2006, p. 512. ISBN: 9781930934191. URL: https://vtk.org/about/.

[94] Cand Scient Thesis and Erlend Hodneland. *Segmentation of Digital Images*. Tech. rep. 2003. URL: https://folk.uib.no/nmaxt/thesis/erlend.pdf.

[95] David W Shattuck et al. "Construction of a 3D probabilistic atlas of human cortical structures." In: *NeuroImage* 39.3 (Feb. 2008), pp. 1064–80. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2007.09.031. URL: http://www.ncbi.nlm.nih.gov/pubmed/18037310%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2757616.

[96] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: (Sept. 2014). URL: http://arxiv.org/abs/1409.1556.

[97] Zhuang Song et al. "Integrated Graph Cuts for Brain MRI Segmentation". In: Springer, Berlin, Heidelberg, 2006, pp. 831–838. DOI: 10.1007/11866763{_}102. URL: http://link.springer.com/10.1007/11866763_102.

[98] T. Sørensen. "A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons". In: *Kongelige Danske Videnskabernes Selskab* 5.4 (1948), pp. 1–34.

[99] Perry. Sprawls. *Magnetic resonance imaging : principles, methods, and techniques*. Medical Physics Pub, 2000, p. 173. ISBN: 9780944838976. URL: http://www.sprawls.org/mripmt/index.html.

[100] Carole H Sudre et al. "Generalised Dice overlap as a deep learning loss function for highly unbalanced segmentations". In: (July 2017). DOI: 10.1007/978-3-319-67558-9{_}28. URL: http://arxiv.org/abs/1707.03237%20http://dx.doi.org/10.1007/978-3-319-67558-9_28.

[101] Tiezhu Sun et al. *Image-level to Pixel-wise Labeling: From Theory to Practice*. Tech. rep. 2017. URL: https://www.ijcai.org/proceedings/2018/0129.pdf.

[102] Christian Szegedy et al. "Going Deeper with Convolutions". In: (Sept. 2014). URL: https://arxiv.org/abs/1409.4842.

[103] Christian Szegedy et al. *Rethinking the Inception Architecture for Computer Vision*. Tech. rep. URL: https://arxiv.org/pdf/1512.00567v3.pdf.

[104] Saeid Asgari Taghanaki et al. "Combo Loss: Handling Input and Output Imbalance in Multi-Organ Segmentation". In: (May 2018). URL: http://arxiv.org/abs/1805.02798.

[105] *TensorFlow*. URL: https://www.tensorflow.org/.

[106] Martin Thoma. "A Survey of Semantic Segmentation". In: (Feb. 2016). URL: http://arxiv.org/abs/1602.06541.

[107] Gijs van Tulder. *Elastic deformations for N-dimensional images (Python, SciPy, NumPy, TensorFlow)*. https://github.com/gvtulder/elasticdeform. 2013.

[108] Catherine. Westbrook. *MRI at a glance*. Blackwell Science, 2002, p. 107. ISBN: 0632056193.

[109] David H. Wolpert. "Stacked generalization". In: *Neural Networks* 5.2 (Jan. 1992), pp. 241–259. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(05)80023-1. URL: https://www.sciencedirect.com/science/article/pii/S0893608005800231.

[110] Saining Xie et al. "Aggregated Residual Transformations for Deep Neural Networks". In: (Nov. 2016). URL: http://arxiv.org/abs/1611.05431.

[111] Rikiya Yamashita et al. “Convolutional neural networks: an overview and application in radiology”. In: *Insights into Imaging* 9.4 (Aug. 2018), pp. 611–629. ISSN: 1869-4101. DOI: 10.1007/s13244-018-0639-9. URL: https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9.

[112] Linjie Yang, Ning Xu, and Adobe Research. *Video Instance Segmentation.* Tech. rep. URL: https://github.com/.

[113] Léon Bottou Yann LeCun and Patrick Haffner Yoshua Bengio. “Gradient-Based Learning Applied to Document Recognition”. In: *IEEE* (1998).

[114] Fei Ye. “Particle swarm optimization-based automatic parameter selection for deep neural networks and its applications in large-scale and high-dimensional data.” In: *PloS one* 12.12 (2017), e0188746. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0188746. URL: http://www.ncbi.nlm.nih.gov/pubmed/29236718%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5728507.

[115] Song Yuheng and Yan Hao. *Image Segmentation Algorithms Overview.* Tech. rep. URL: https://arxiv.org/ftp/arxiv/papers/1707/1707.02051.pdf.

[116] Matthew D. Zeiler. “ADADELTA: An Adaptive Learning Rate Method”. In: (Dec. 2012). URL: http://arxiv.org/abs/1212.5701.

[117] Zhaoye Zhou et al. “Deep convolutional neural network for segmentation of knee joint anatomy”. In: *Magnetic Resonance in Medicine* 80.6 (Dec. 2018), pp. 2759–2770. ISSN: 15222594. DOI: 10.1002/mrm.27229. URL: http://www.ncbi.nlm.nih.gov/pubmed/29774599%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC6342268%20http://doi.wiley.com/10.1002/mrm.27229.

[118] Zongwei Zhou et al. “UNet++: A Nested U-Net Architecture for Medical Image Segmentation”. In: (July 2018). URL: http://arxiv.org/abs/1807.10165.

[119] A P Zijdenbos et al. “Morphometric analysis of white matter lesions in MR images: method and validation.” In: *IEEE transactions on medical imaging* 13.4 (1994), pp. 716–24. ISSN: 0278-0062. DOI: 10.1109/42.363096. URL: http://www.ncbi.nlm.nih.gov/pubmed/18218550.

Chapter 9: Appendix

9.1 Experiment 2D U-Nets

Some of the initial experimentation for this thesis was conducted using 2D U-Nets. The implemented networks followed the same structure as the original U-Net. The initial and naive approach was feeding the network with slices taken from the sagittal plane with a resolution of 400×400. However, the U-Net was only able to learn to segment the bone segment class.

The second approach was to split the segmentation task into three different models, one for each segment class cropping out the region each segment class occupies. By this approach, the networks managed only sometimes to learn to segment the ACL and PCL classes. The segmentation by 2D U-Nets compared to segmentation by 3D U-Nets for this specific segmentation problem was quite bad in comparison. Because of time limitations and the superior performance of the 3D, U-Nets the use of 2D networks was not further explored.

9.2 GUI images

As discussed in section 4.7 the 3D models can be exported and be used in 3D software. Here we present how the 3D models can be used for various applications.

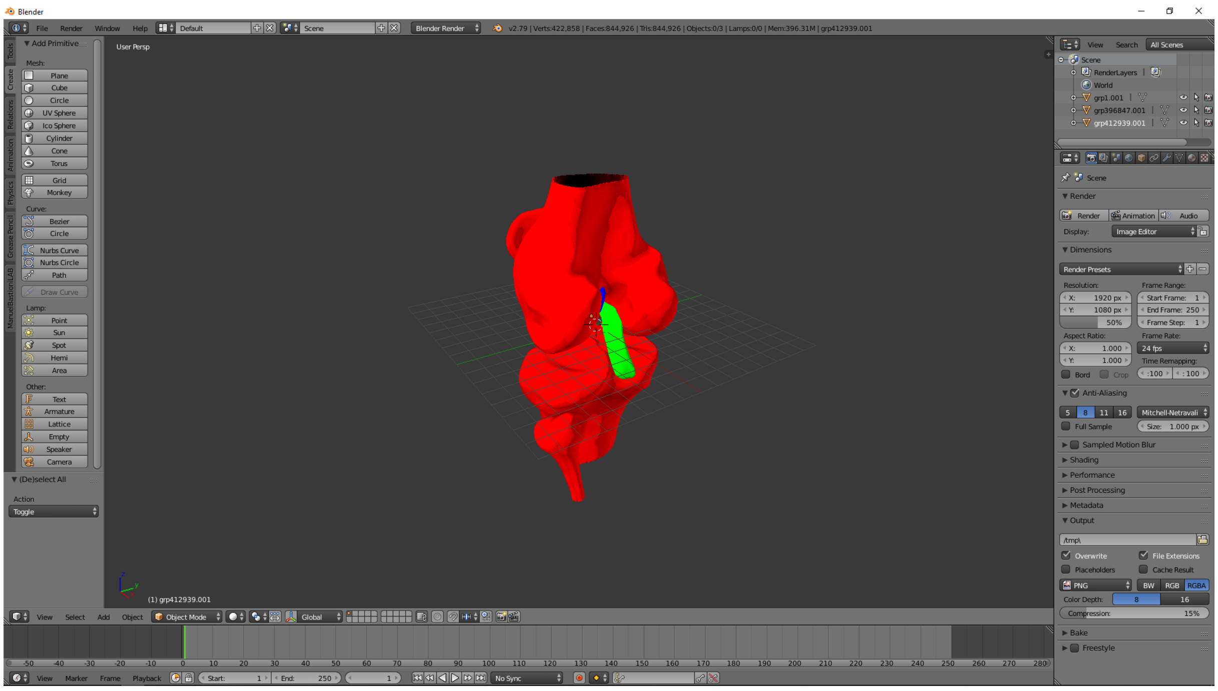
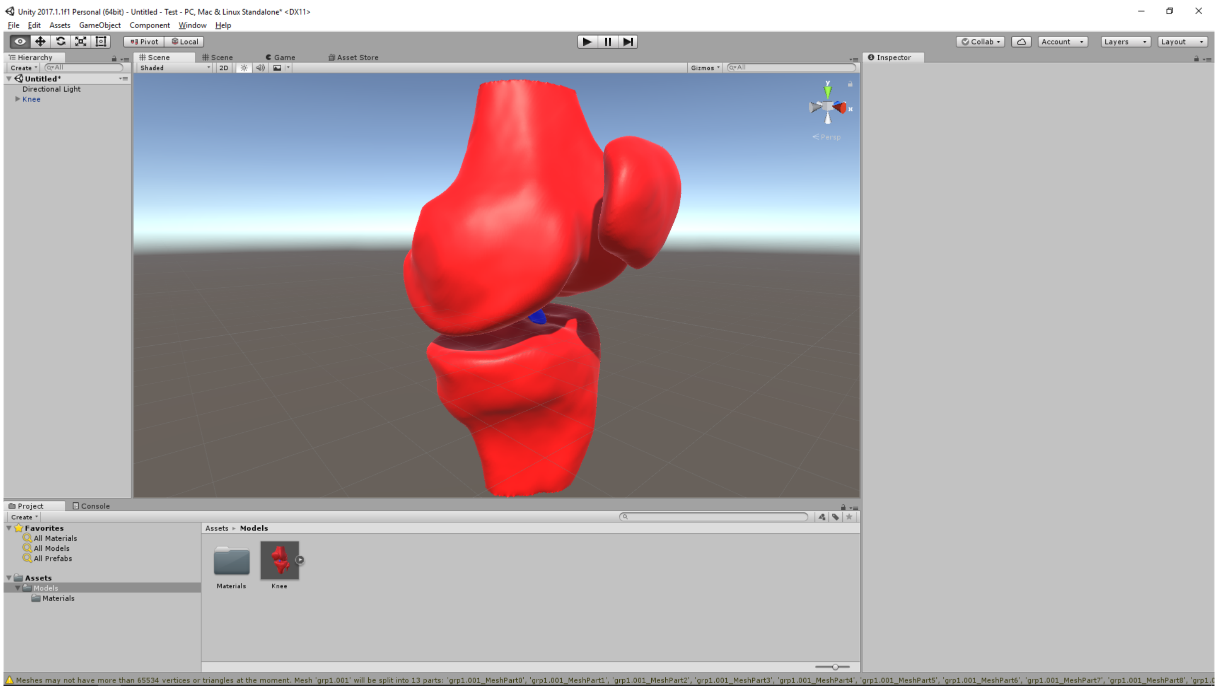

Blender
v2.79 | Verts:422,858 | Faces:844,926 | Tris:844,926 | Objects:0/3 | Lamps:0/0 | Mem:396.31M | grp412939.001
User Persp
(1) grp412939.001
grp1.001
grp396847.001
grp412939.001

Figure 9.1: 3D model exported to wavefront format and edited in Blender.



Meshes may not have more than 65534 vertices or triangles at the moment. Mesh 'grp1.001' will be split into 13 parts: 'grp1.001_MeshPart0', 'grp1.001_MeshPart1', 'grp1.001_MeshPart2', 'grp1.001_MeshPart3', 'grp1.001_MeshPart4', 'grp1.001_MeshPart5', 'grp1.001_MeshPart6', 'grp1.001_MeshPart7', 'grp1.001_MeshPart8', 'grp1.0

Figure 9.2: 3D model exported to .fbx format from blender and imported to Unity.

The 3D models could also be exported to the .stl format and be 3D printed. For the .stl format the 3D model is split into 3 different models for the segment classes. The following images illustrate the exported .stl models ready to be printed.

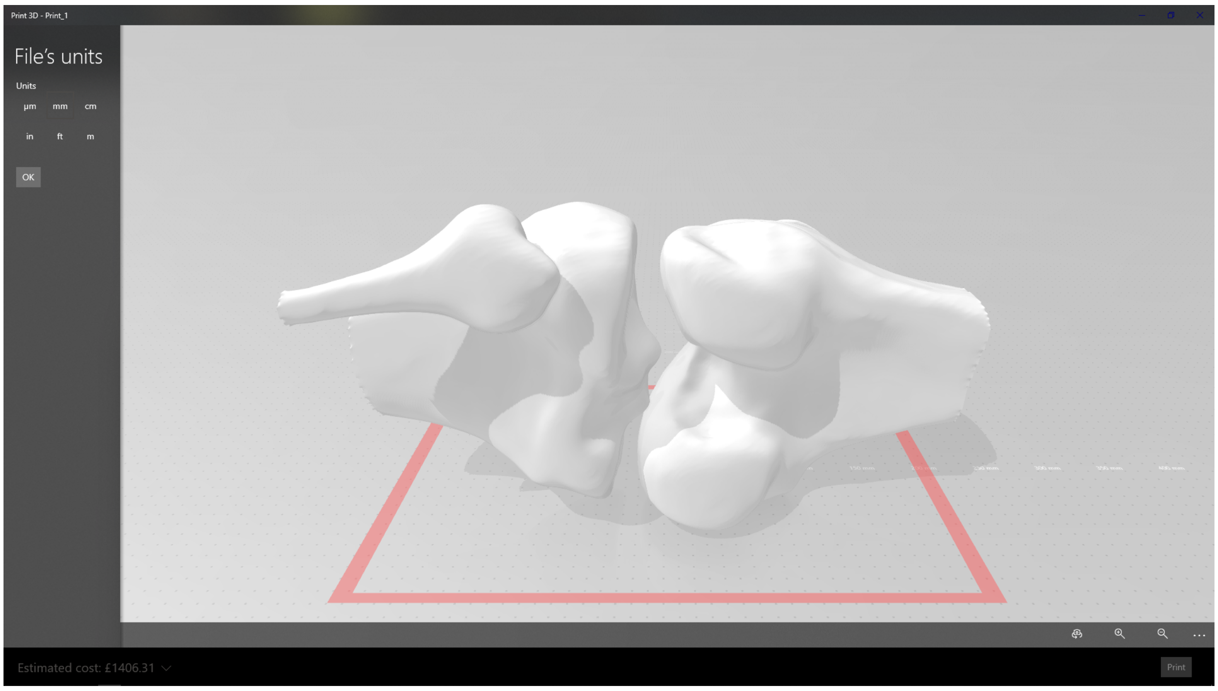


Figure 9.3: 3D model of the bone exported to .stl format and opened in Microsofts 3D printing software

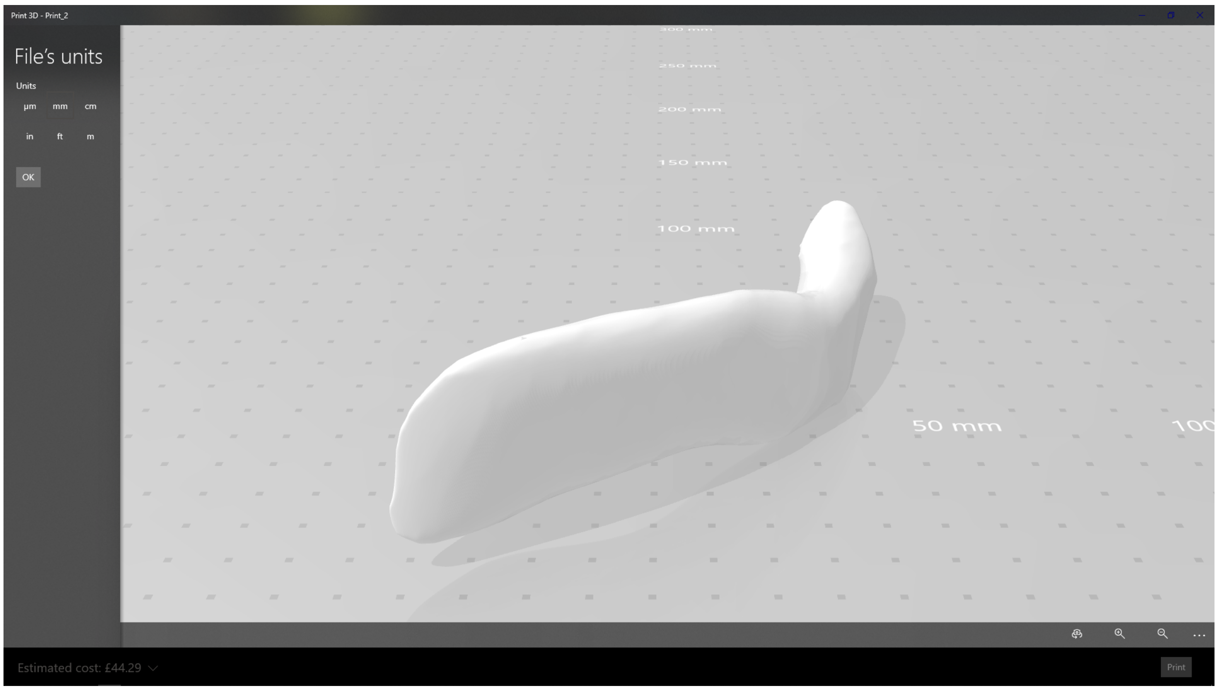


Figure 9.4: 3D model of the PCL exported to .stl format and opened in Microsofts 3D printing software

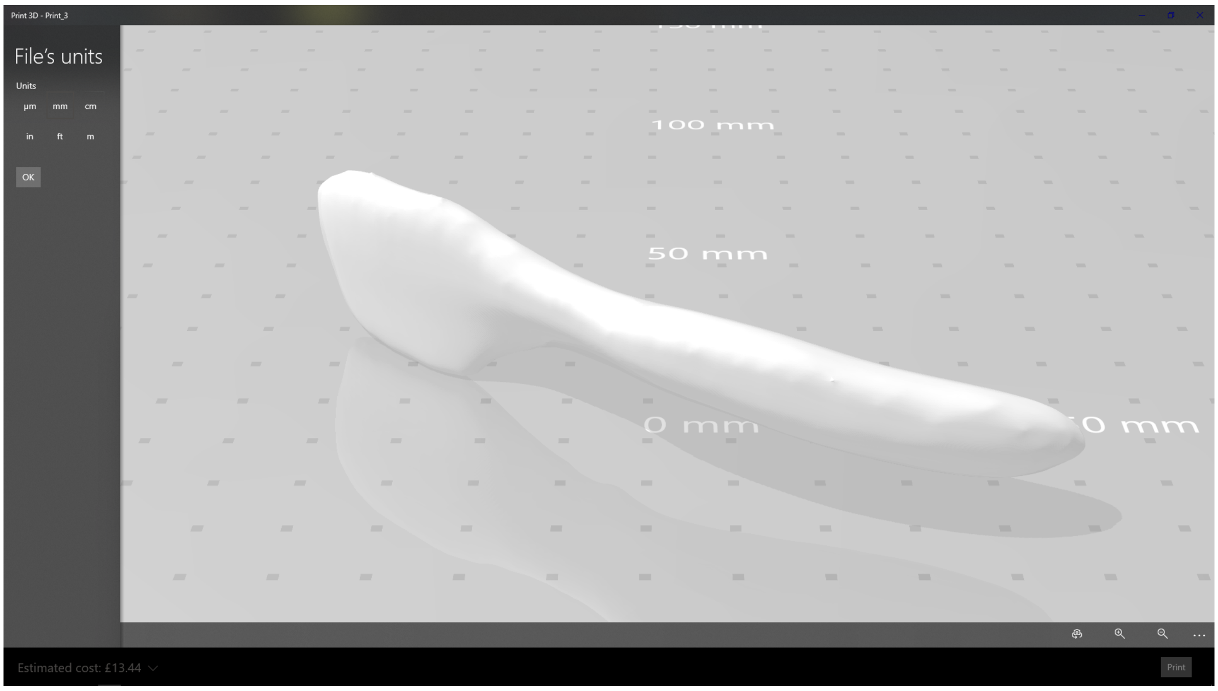
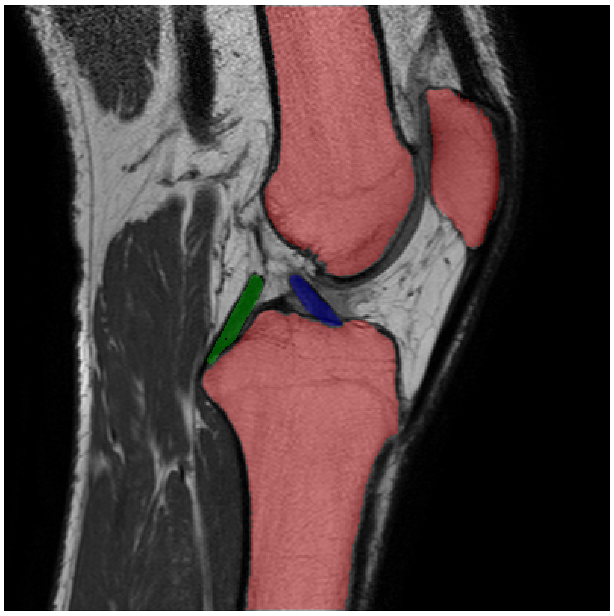


Figure 9.5: 3D model of the ACL exported to .stl format and opened in Microsofts 3D printing software

9.3 Alignment issue


(a) Ground truth

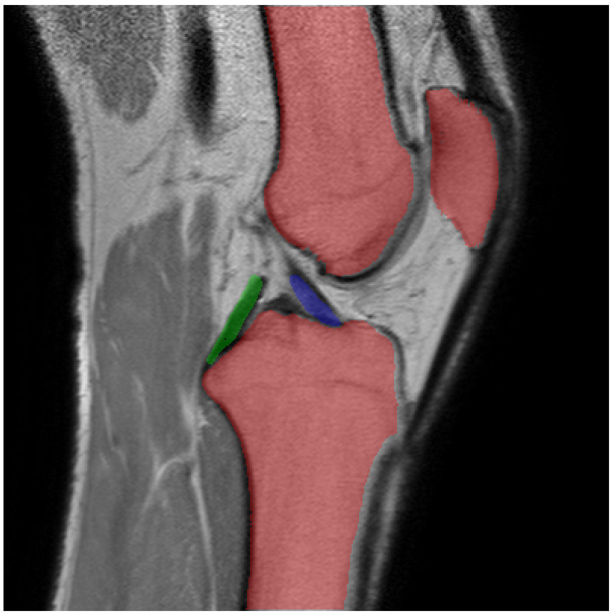
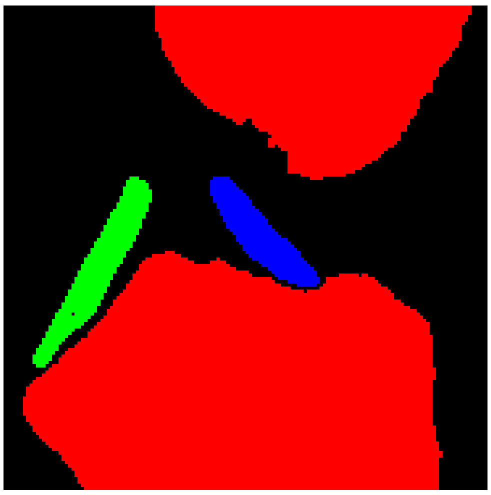
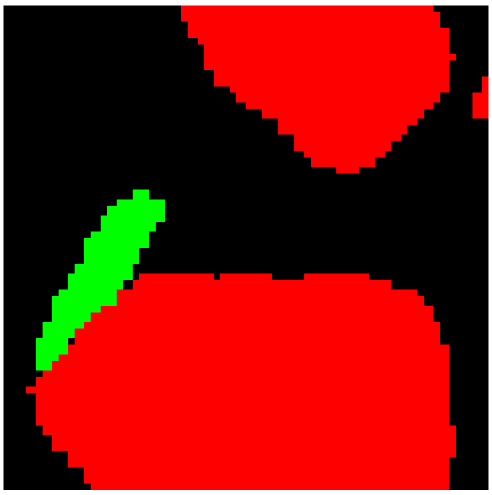
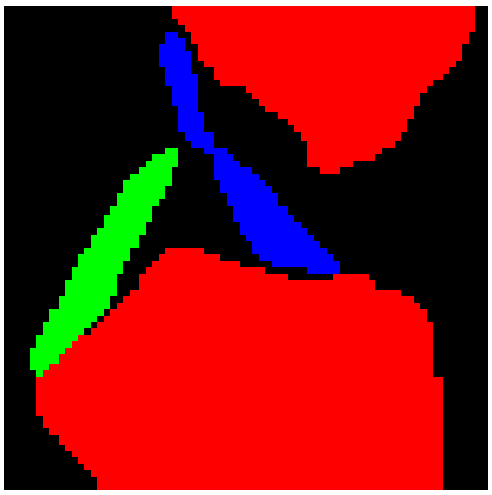

(b) FS 1

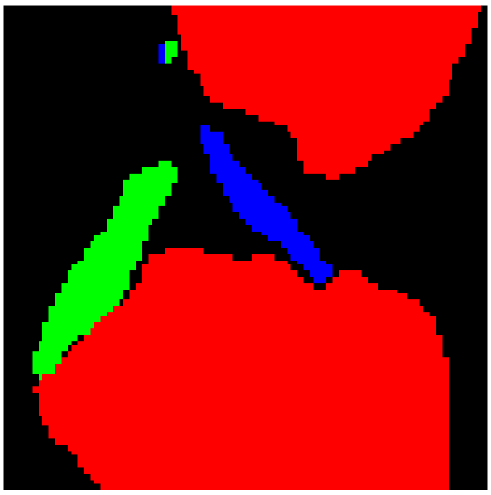
Figure 9.6: Example showing the alignment issue for corresponding T1 and PD volume.

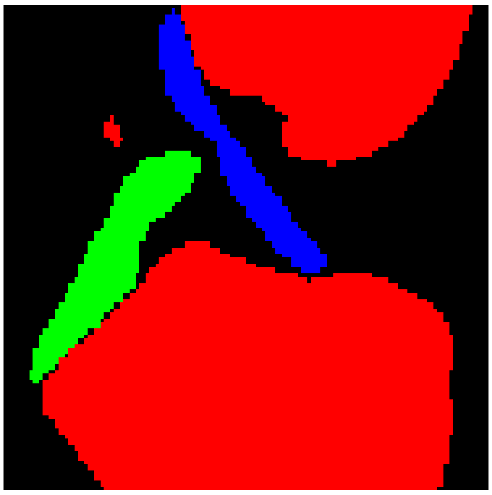
9.4 Resolution experiment images


(a) Ground truth


(b) FS 1


(c) FS 2


(d) FS 3


(e) FS 4

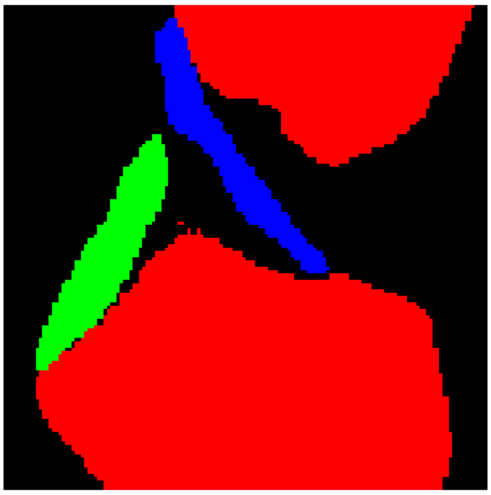
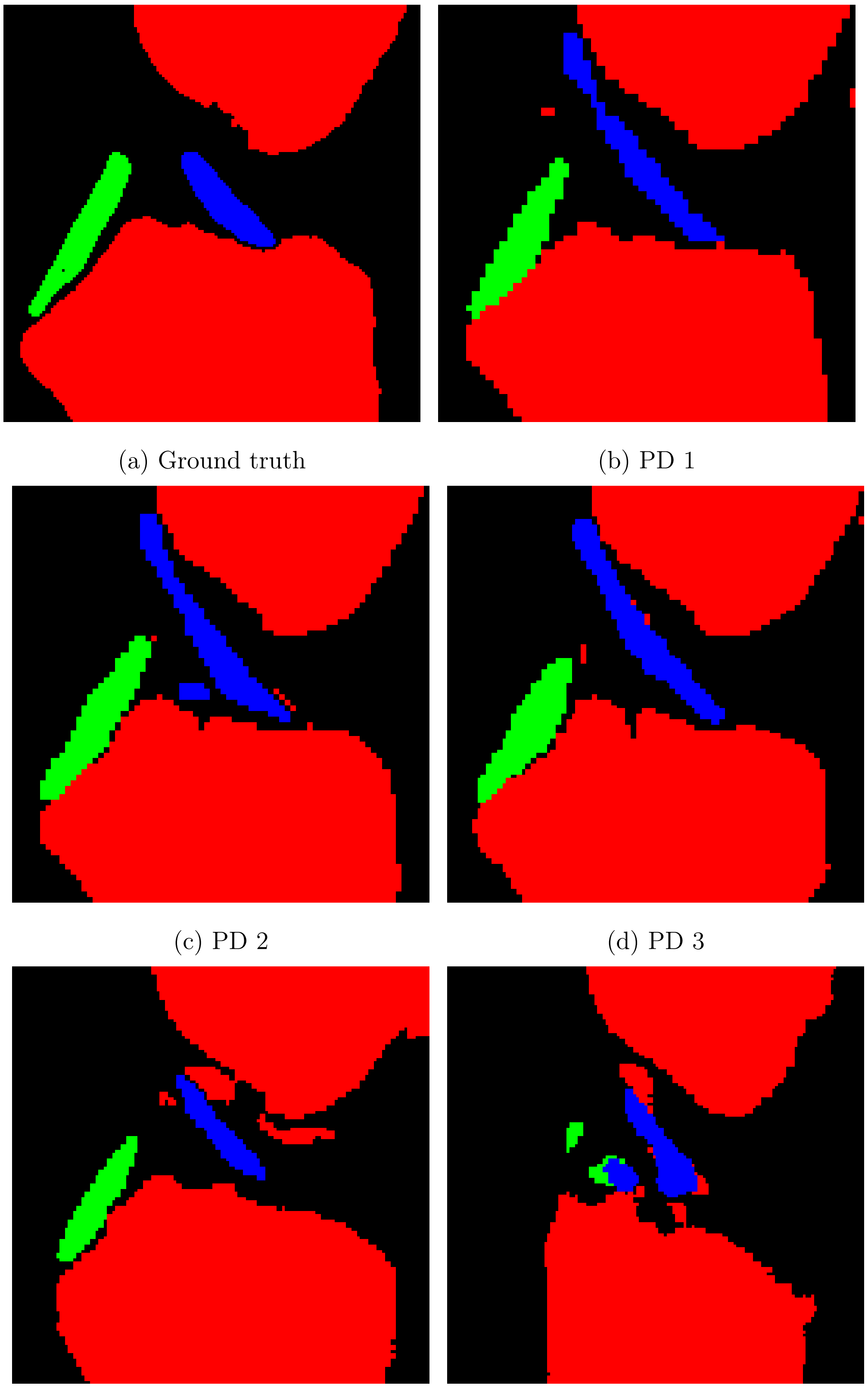

(f) FS 5

Figure 9.7: Zoomed in example showing output from FS models trained using different input resolution.

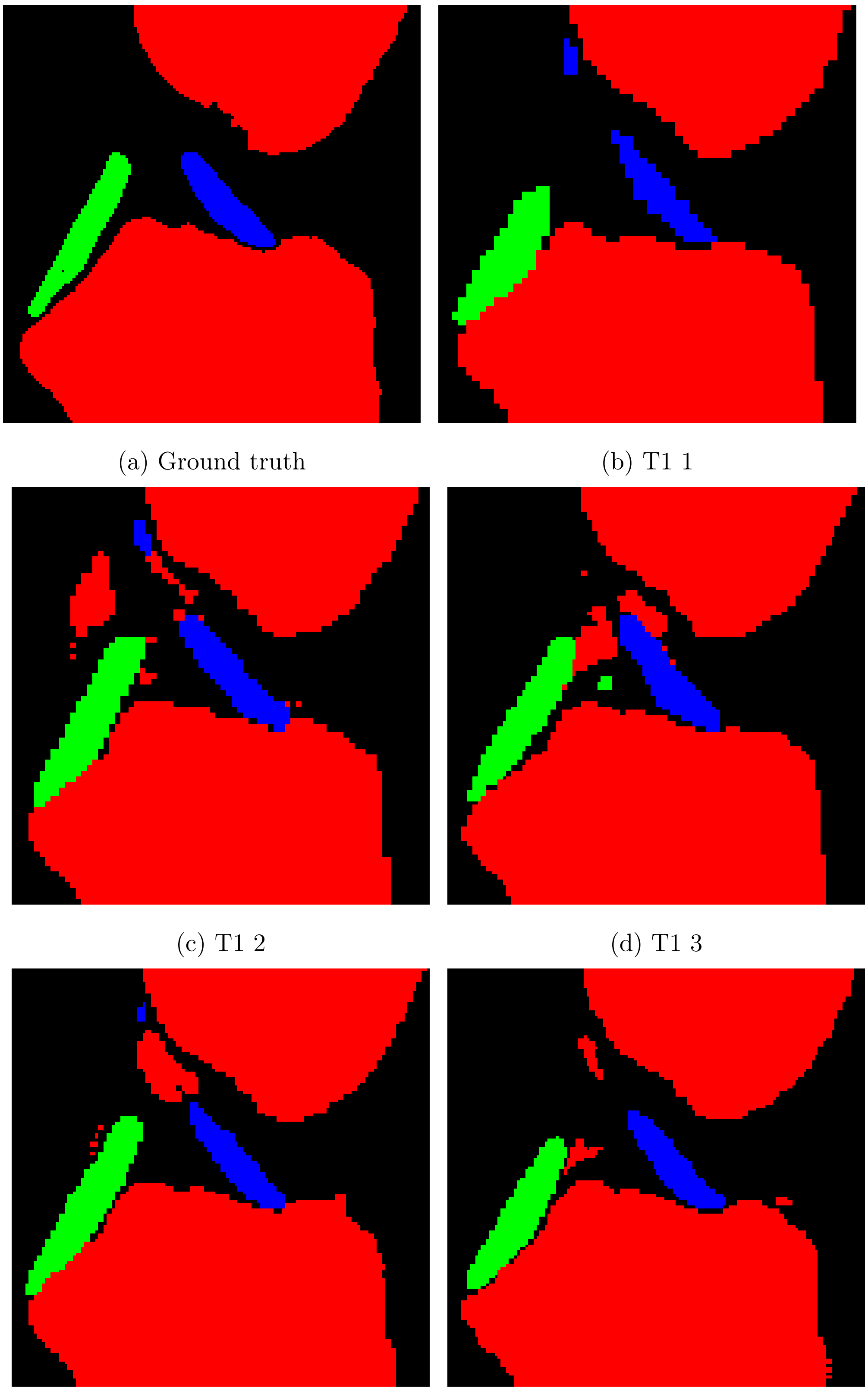


(a) Ground truth (b) PD 1

(c) PD 2 (d) PD 3

(e) PD 4 (f) PD 5

Figure 9.8: Zoomed in example showing output from PD models trained using different input resolution.



(a) Ground truth

(b) T1 1

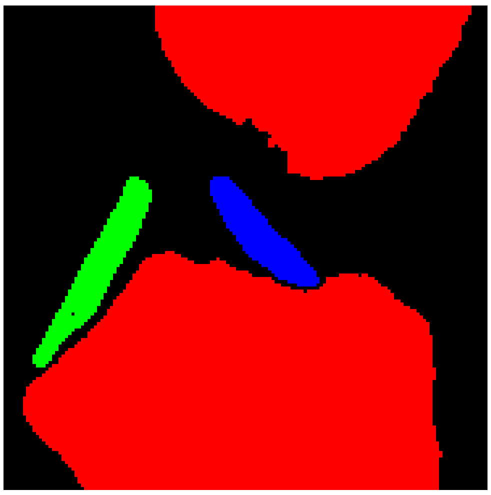
(c) T1 2

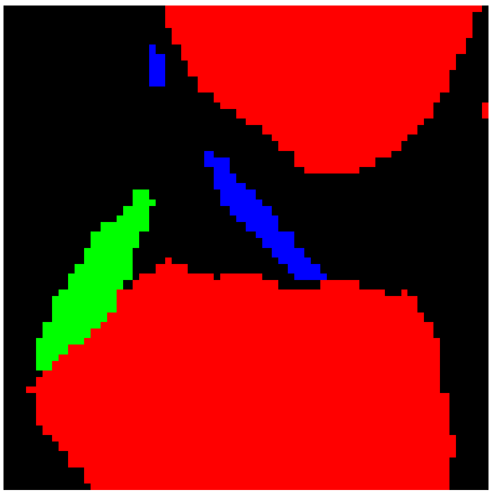
(d) T1 3

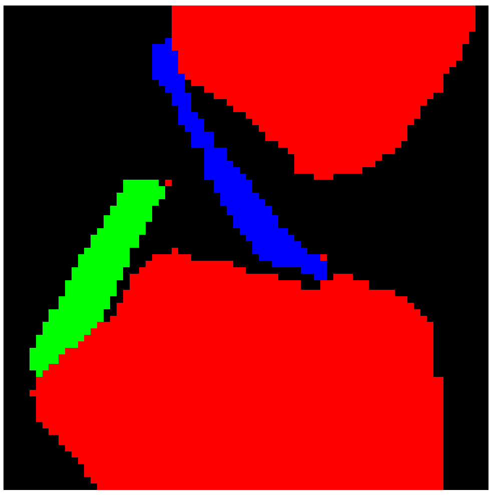
(e) T1 4

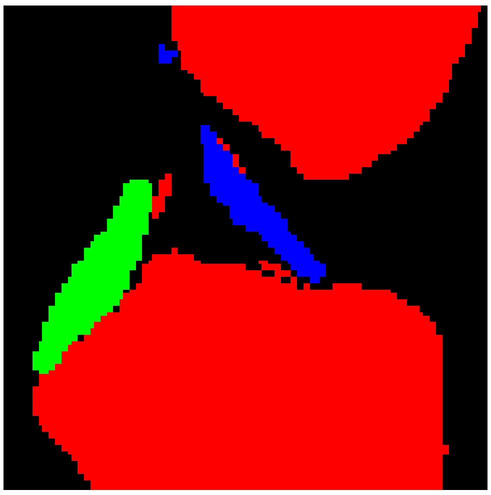
(f) T1 5

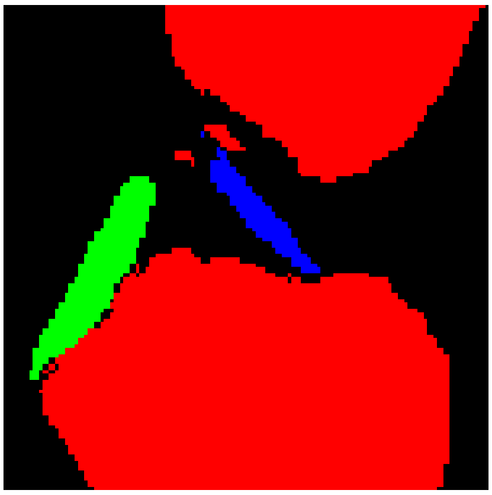
Figure 9.9: Zoomed in example showing output from T1 models trained using different input resolution.


(a) Ground truth


(b) Ensemble 1


(c) Ensemble 2


(d) Ensemble 3


(e) Ensemble 4

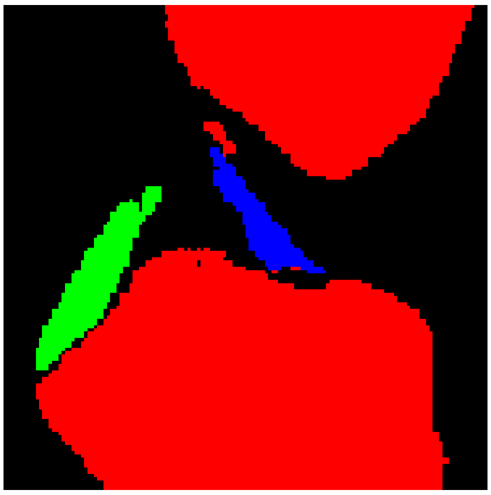
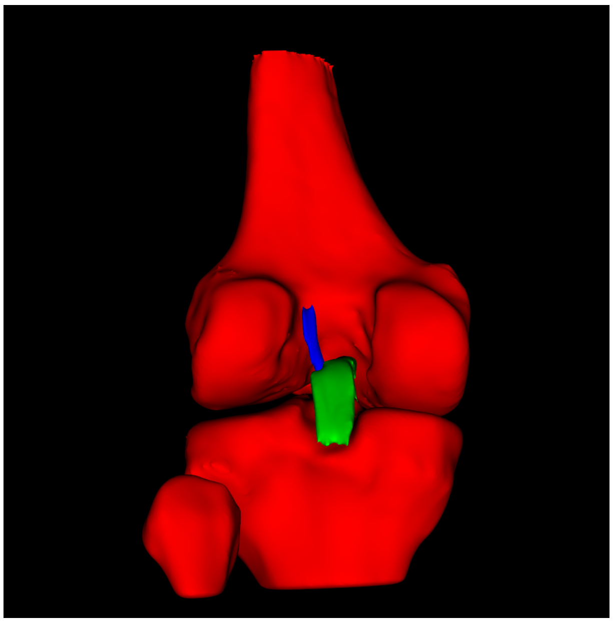

(f) Ensemble 5

Figure 9.10: Zoomed in example showing output from ensemble models trained using different input resolution.

9.5 Final fragmented model comparison images


(a) Ground truth mask

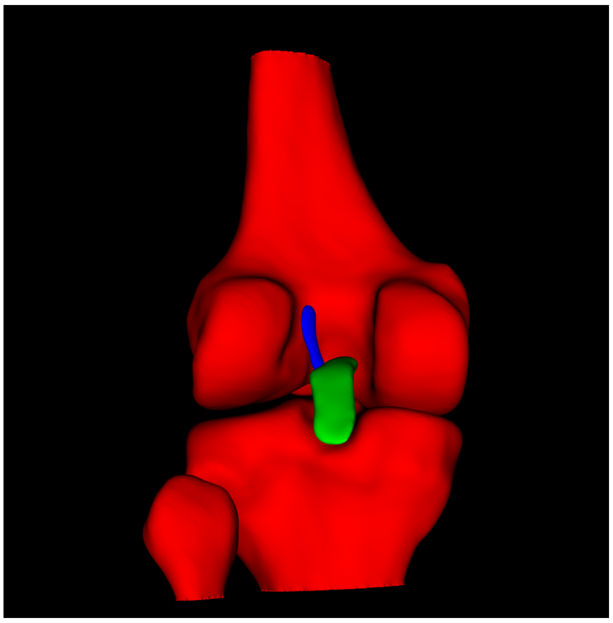
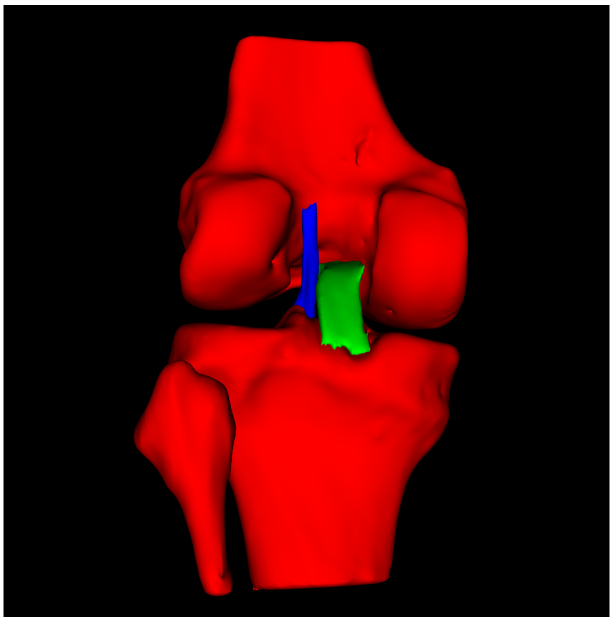

(b) Fragmented model prediction mask

Figure 9.11: Comparison DL005


(a) Ground truth mask

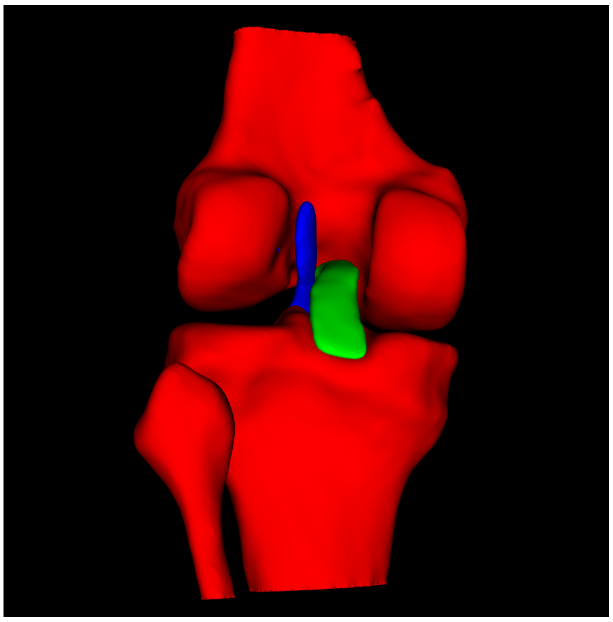
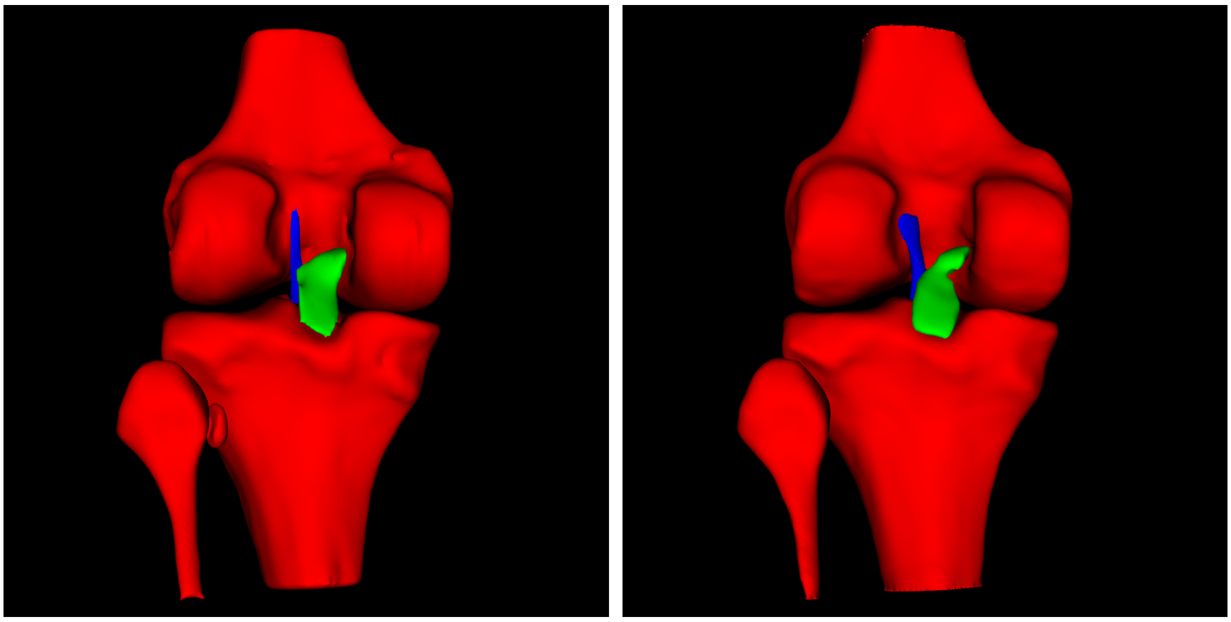
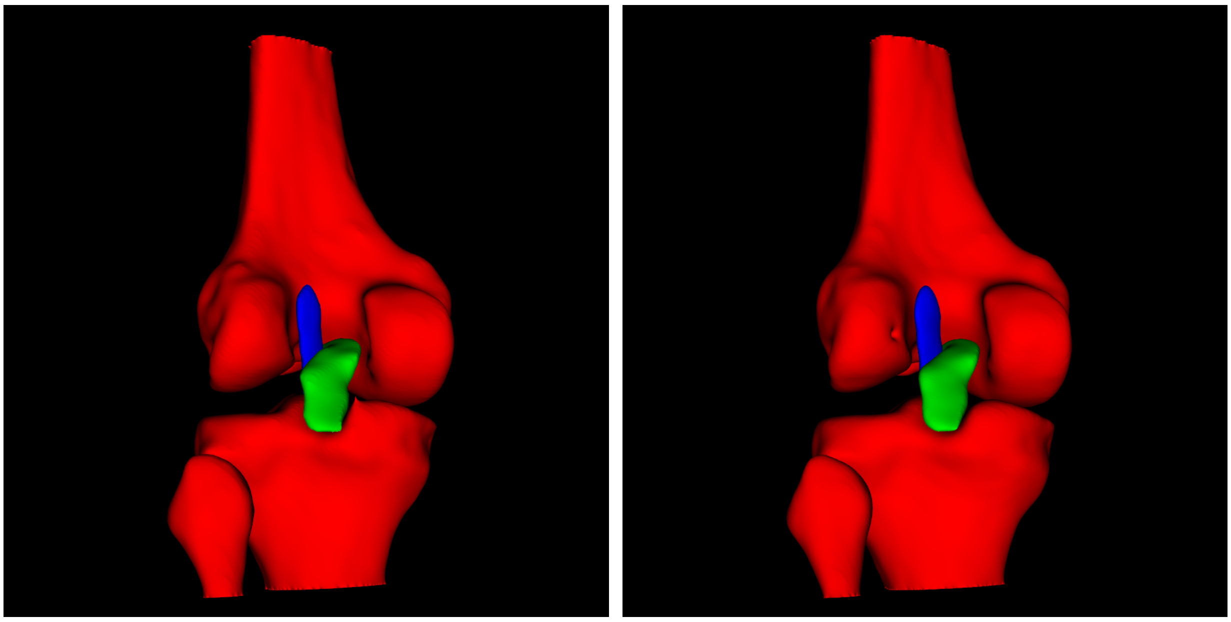

(b) Fragmented model prediction mask

Figure 9.12: Comparison DL006



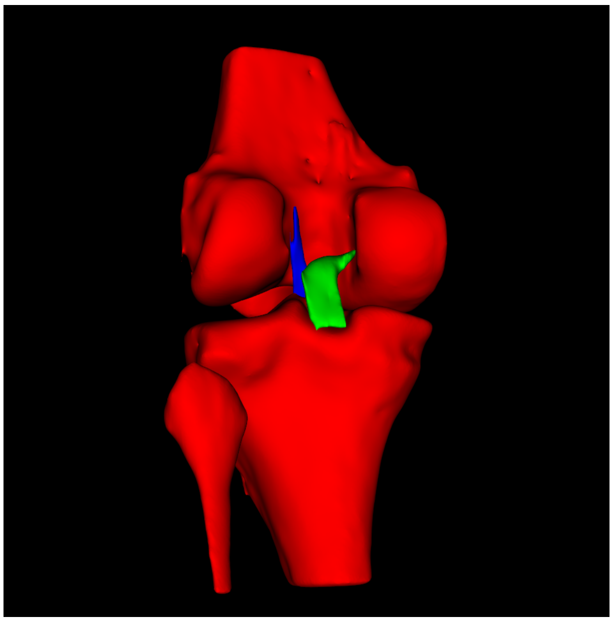
(a) Ground truth mask (b) Fragmented model prediction mask

Figure 9.13: Comparison DL007



(a) Ground truth mask (b) Fragmented model prediction mask

Figure 9.14: Comparison DL015


(a) Ground truth mask

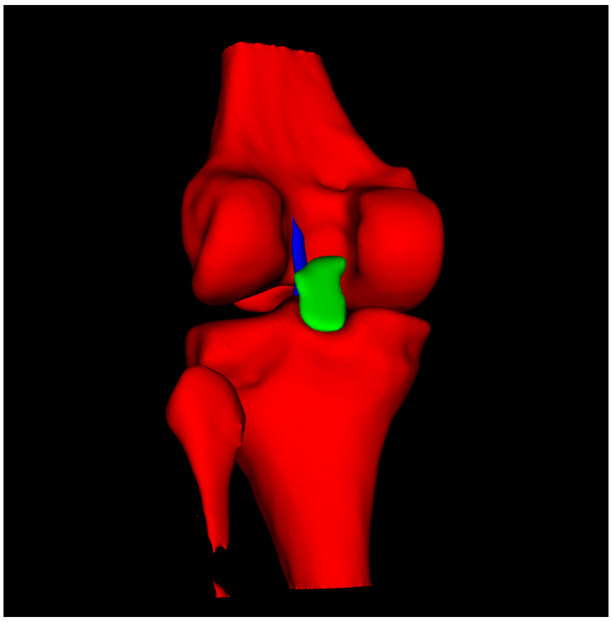
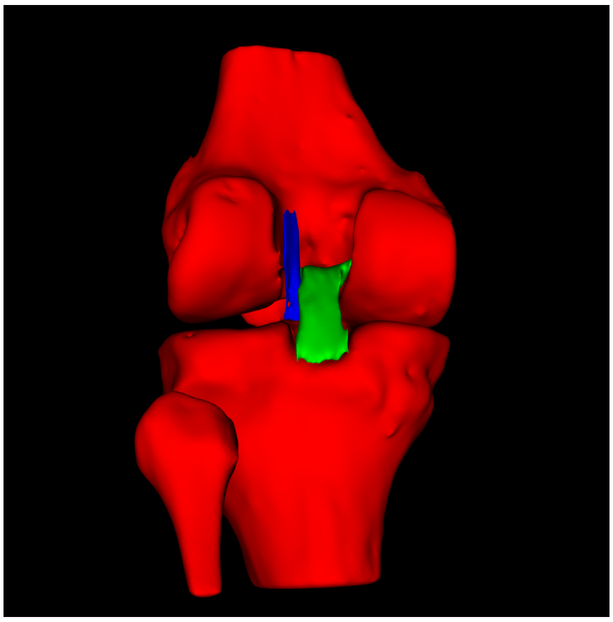

(b) Fragmented model prediction mask

Figure 9.15: Comparison DL016


(a) Ground truth mask

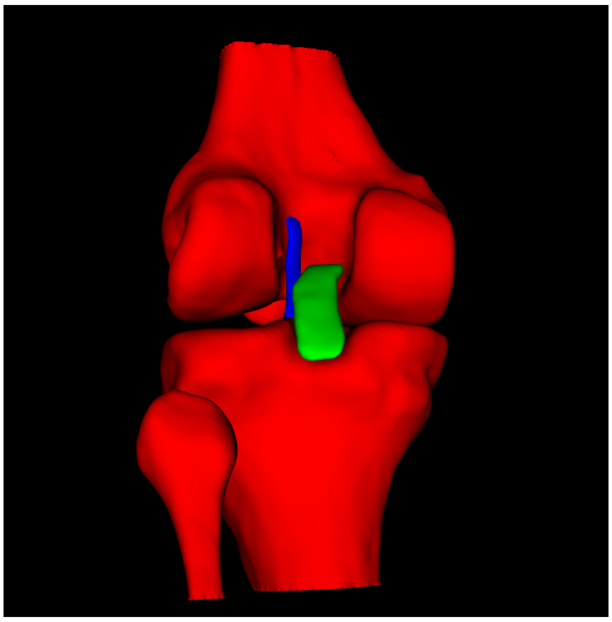
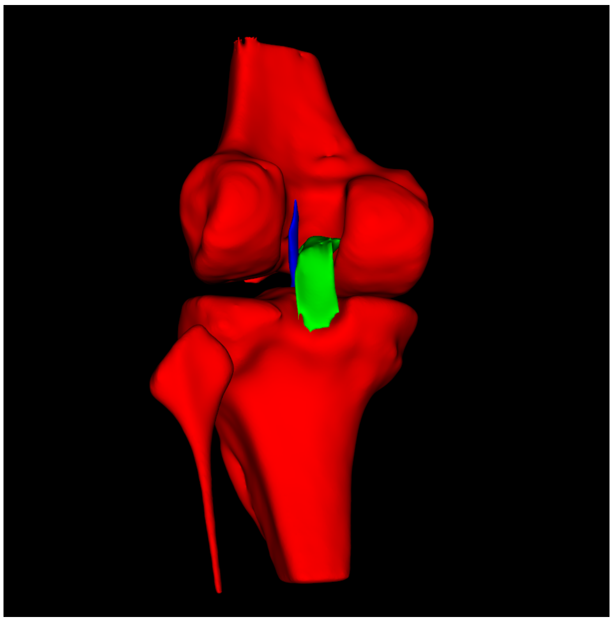

(b) Fragmented model prediction mask

Figure 9.16: Comparison DL017


(a) Ground truth mask

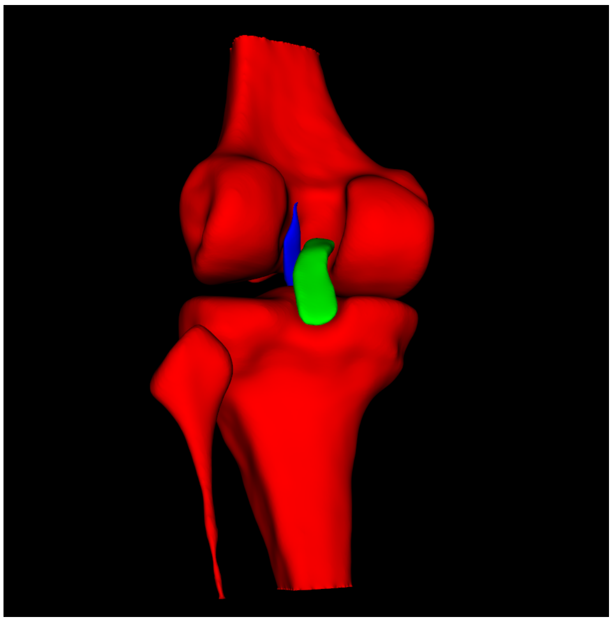

(b) Fragmented model prediction mask

Figure 9.17: Comparison DL018