FACULTY OF SCIENCES

Spine vertebrae segmentation in 3D CT scan images

Jan Alexander
Student number: 00604831

Supervisor: Dr. Joris Roels
Counsellor: Dr. Bert Vankeirsbilck (Ugent - IMEC)

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Statistical Data Analysis

Academic year 2020-2021

GHENT UNIVERSITY

Admission for circulating the work

Foreword

Before you lies my master thesis, "Spine vertebrae segmentation in 3D CT scan images". This document presents my investigation of the weakly supervised segmentation problem applied on medical scans of the human lumbar spine. I hope to convince the reader of the usefulness of the two loss function components I devised for this problem and the algorithm I developed to combine three segmentation models of two-dimensional slices of the volumes to produce an improved segmentation result.

I have developed this research question with my supervisors, Dr J. Roels and Dr B. Vankeirsbilck, who encouraged me to orient my thesis towards weakly supervised problems.

I want to thank my supervisors for their guidance and support during researching this topic and developing the procedure described in this document. All the more, given the extraordinary conditions we have all experienced in the past two years.

I also want to thank my colleagues and coordinators at Verhaert for their moral support during the last two years. The combination of a job and academic studies is a demanding undertaking. It was rewarding to find the skills I studied at night directly applicable to my day-time job at times.

My parents deserve a particular note of thanks for their support and kind words. These have, as always, served me well.

I hope you enjoy the reading,
Jan Alexander

Gent, August 2021

Contents

Acronyms

ANN . . . . . Artificial Neural Network

BP . . . . . Backpropagation

CLAHE . . . . . Contrast Limited Adaptive Histogram Equalization

CNN . . . . . Convolutional Neural Network

CT . . . . . Computer Tomography

FCN . . . . . Fully Convolutional Neural Network

HMM . . . . . Hidden Markov Model

IoU . . . . . Intersection over Union

LC . . . . . Location based Counting loss

MIL . . . . . Multi-Instance Learning

ML . . . . . Machine Learning

MRI . . . . . Magnetic Resonance Imaging

NLP . . . . . Natural Language Processing

OSF . . . . . Open Science Foundation

PCAM . . . . . Point supervised Class Activation Map

PPG . . . . . Photoplethysmogram

RGB . . . . . Red, Green, Blue images (common digital camera colour images)

RNN . . . . . Recurrent Neural Network

RoI . . . . . Region of Interest

US . . . . . Ultra Sound imaging

Glossary

Artificial Intelligence

The study of using computers to automatically perform tasks which once were considered only humans could do. This includes, but is not restricted to, the interpretation of speech and images. It is often referred to with the acronym AI. . x, 6, 10, 11, 16

Class Activation Map

Technique to identify region of an image *responsible* for the classification result. The class activation map indicates the discriminative elements of an image that cause the CNN to identify the category. Several techniques aim to do this, of which the CAM technique can be considered the most popular one. The CAM technique is based on the feature maps obtained in the last convolution layer.. x, 19

Computated Tomography

CT is the imaging of a volume through the use of a penetrating wave X-ray wave. Through these waves, a collection of images, called *tomograms*, are produced. The mathematical procedure to reconstruct the original volume based on these images is called *tomographic reconstruction*. A Computer Tomography (CT) scan is produced through tomographic reconstruction of several X-ray radiographs. . x, 5

COVID-19

Corona virus disease '19. The disease caused by the SARS-CoV-2 virus.. x, 20

Deep Learning

Deep learning is a branch of Machine Learning where a set of multiple sequential layers is used to extract higher-level features from the raw input data progressively. Different deep learning architectures have proven to allow the construction of well-performing models for problems where other machine learning techniques seem to be less performant such as computer vision (with convolution layers) and natural language processing (with recurrent networks such as LSTM). The key to the success of deep learning networks is the ability to automatically perform feature extraction in the first layers without requiring human guidance. . x, 6, 11

features

A feature is a property of a phenomenon. For all machine learning tasks, it is crucial to obtain informative, discriminative and independent features. The transformation extracting informative and non-redundant features from the original measured (*raw*) data is the *feature extraction* step. The feature set is crucial in the subsequent learning or inference step. In deep learning applications, the feature set (specifically the feature set resulting from a contracting path) is often called the *encoding.*. x, 8, 12, 33

ground truth

The *Ground Truth* is a term used in machine learning to indicate the ideal expected result. In the context of Instance Segmentation, the ground truth is the actual class of every pixel or voxel. . x, 12, 37, 52, 54

Houndsfield Units

Unit of attenuation of X-ray waves. The radiodensity of air is defined as $HU = -1000$, while the radiodensity of destilled water is $HU = 0$. This unit is used for Computer Tomography (CT) scan images. . x

Machine vision

The branch of Artificial Intelligence to automatically obtain information from images. In this work, these images can be both two dimensional (*pictures*) as three dimensional (*volumes*). . x, 1, 6, 11, 16

ResNet

A residual network is a neural network architecture. It can be very deep due to the skip connection architecture.. x, 34

Segmentation

The *segmentation* problem in machine vision consists of the classification of each pixel or voxel. The problem of *semantic* segmentation is to detect, for each pixel, the object category to which it belongs. *Instance* segmentation digs deeper. It identifies for each pixel the object instance to which it belongs. The difference is that it differentiates between two objects of the same object category in the picture. . x, 11

supervised

(Fully) Supervised Machine Learning task where target labels are present. The objective of these problems is to model the relationship between an *input* and an *output*. . x, 14, 21, 40

unsupervised

In an *Unsupervised* machine learning problem, no labels are present. The aim is not to model the relationship between an *input* and an *output*. The aim is to model the structure of the data. Frequent applications of these techniques are clustering and dimensionality reduction of data. . x, 21, 40

weakly supervised

Weakly supervised machine learning where the ground truth labels are only partially available. In image segmentation, this can mean that the labels are only provided at the image level or that the image's point-level annotation is provided. The model is trained on incomplete, noisy or limited labels. The desired result remains the complete segmentation of the image. Just like in the case of *Fully Supervised Learning* the objective is to model the relationship between an *input* and an *output*. Due to the labels being incomplete, there is a more substantial need to identify the internal structure of the data, as is the case for *Unsupervised learning*. . x, 1, 6, 12, 16, 18, 20, 29

Submission and Formatting Instructions for International Conference on Machine Learning (ICML 2021)

Jan Alexander [1] **Joris Roels** [2] **Bert Vankeirsbilck** [3]

Abstract

Medical professionals use mri or ct scans as essential components for medical diagnosis, following the course of medical conditions and the planning of medical procedures. There is a trend towards machine vision to support medical professionals interpreting and using these images. Building these applications requires expensive labelled datasets. This research investigates techniques to reduce the dataset labelling cost by working with point annotation instead of full annotation. Experiments are conducted on publicly available datasets and demonstrate two new loss components and a combination technique of different model results to generate pseudo masks. As a final result, this work demonstrates that one can obtain 72 % of the performance of a fully annotated model at an estimated 12 % of the labelling cost.

1. Thesis objective & Motivation

The use of radiological images is a crucial element in modern medical practice. mri or ct scans are essential components for pre-operative and post-operative diagnosis, following the course of medical conditions and the planning of medical procedures. Automated interpretation of medical images can mean a gain in efficiency.

Machine vision - deep learning in general - tends to be very *data-hungry*. Constructing a new model requires large, labelled datasets. Acquiring these datasets and the corresponding labels is time-consuming and expensive. Maximisation of the return of a given data and labelling budget through is a goal shared by all ml practitioners. The use of weak labels, or sometimes called *hints*, is one approach to attempt this. This approach aims to train a model capable of inferring more informative results than the information level explicitly available in the labelling.

[1]Master Statistical Data Analysis [2]UGent, VIB [3]UGent, IMEC. Correspondence to: Jan Alexander <jan.alexander@ugent.be>.

Abstract master thesis Jan Alexander

This project presents a model for the automated segmentation of the lumbar vertebrae of the human spine based on point level annotated medical scans. Point level annotation is faster and cheaper than providing a complete label mask (estimated at 12% of cost(**?**)), this technique provides a cost-benefit. The labels only contain the true class of a mere handful of voxels. This is a weak label to classify all voxels.

2. Data sets and data preprocessing

All datasets used in this work are publicly available (all datasets are listed on page **??**). These datasets contain both ct and mri scans. In 86 of these scans, complete volume masks of the vertebrae are available. In 20 volumes, only semantic labels are available. For 125 volumes, point level annotation is available.

The complete dataset of 231 patients consists of 112 women and 99 men. Of 23 people, no gender information is available. Since a medical professional does not order a medical scan unless there is a suspicion of a medical condition, the dataset contains various patients with different pathologies, such as patients with scoliosis and with crushed and wedged vertebrae.

Different datasets vary in data formats and different scan resolutions. Data preprocessing starts with homogenising the scan resolution by resampling the image on an $1mm \times 1mm \times 1mm$ grid. Next, the image is sliced along one of the three principal axes. The contrast of the 2D image slices is first enhanced with the clahe algorithm. Then the images are cropped (or padded, if needed) to form $352px \times 352px$ slices. All models are built with this image size, sufficient to contain all 5 lumbar vertebrae L_1 to L_5 in one image.

3. Methodology

The performances of different models are compared based on the class-weighted dice score. This metric takes into account both the model precision and recall as well as the class imbalance.

For 86 scans, full annotation masks are available. As a performance benchmark, the performance of a fully supervised

model trained on these images ($Dice_w = 0,76$) is taken.

3.1. Weakly supervised models

The model backbond is the VGG16-FCN8 network, pretrained on a large classification dataset. The model estimates 6 segmentation classes (5 lumbar vertebrae and the background class). By training three different weakly supervised models on sets of 2D images sliced along the 3 main volume dimensions, three sets of segmentation masks are obtained. The combination of these different segmentation masks is used as an *pseudo* label set to train a fully supervised model on one volumetric dimension.

3.1.1. Loss function

To train the weakly supervised network, several loss components, both supervised and unsupervised, are combined. The model loss to train three point-supervised models in the first step of the procedure presented in this work consistents of 4 components: the point loss $\mathcal{L}_P$ and the consistency loss $\mathcal{L}_C$ were defined in (**?**) by I. Laradji, while this work introduced the prior extend and separation loss components $\mathcal{L}_E$ and $\mathcal{L}_S$ are introduced in this work.

The weighted cross-entropy loss is optimised for the fully supervised reference model, a classic choice for this problem. It is also the point loss $\mathcal{L}_P$ component of the weakly supervised model. Then it is only evaluated on the set of available point labels $\mathcal{I}_i$. The function combines the six network output channels with a softmax function σ, after which the negative log-loss function is calculated, weighted with factors w.

$$\mathcal{L}_P(X_i) = -\sum_{\vec{p}\in\mathcal{I}_i} w_{\mathcal{Y}_i(\vec{p})} . \log\left[\sigma_{\mathcal{Y}_i(\vec{p})}\left(z_i(\vec{p})\right)\right] \quad (1)$$

The unsupervised rotation consistency loss $\mathcal{L}_C$ imposes that the model output f_θ should be consistent for a transformation t_k of the input image. In this work, the chosen transformations are image rotations over $0°, 90°, 180°$ or $270°$, combined with an image flip.

$$\mathcal{L}_C(X_i) = \sum_{p\in\mathcal{P}_i} \left| t_k\left[f_\theta(X_i)\right]_p - f_\theta\left(t_k[X_i]\right)_p \right| \quad (2)$$

The second unsupervised loss term is the separation loss term. Due to the low volume of labelled voxels, the the model lacks the incentive to output differentiating expressions of the output channels $\vec{z}_i$. $\mathcal{L}_S$ forces the model to do this.

$$\mathcal{L}_S(X_i) = -\sum_{\vec{p}}\sum_{m\in\mathcal{K}}\sum_{n\in\mathcal{K},n>m} \mathbf{S}(z_i[m]) - \mathbf{S}(z_i[n]) \quad (3)$$

Finally, $\mathcal{L}_E$, the maximal extend supervised loss term, takes into account that a lumbar vertebra has a limited size ($r = 110mm$). The Euclidian distance field $\mathbf{d}$ from the annotation point is converted to a semi-mask for each class k:

$$\mathbf{d}_k(\vec{q}) = \max_{\vec{p}:\mathcal{Y}_i(\vec{p})=k} ||\vec{q}-\vec{p}|| \quad (4)$$

$$\mathbf{m}_k(\vec{q}) = \mathbf{I}\left((-\mathbf{d}(\vec{q})+r) > 0\right) \quad (5)$$

Now, $\mathbf{m}$ is 1 only for positions closer than distance r from the annotation points for class k. Where $\mathbf{m}_k = 0$, the model output should not indicate output class k. Where $\mathbf{m}_k = 1$, the output class is unknown. The loss function is the binary cross-entropy between $\mathbf{m}_k$ and the sigmoid of the kth channel of the logits z_i with weight vector $\{1, 0\}$.

$$\mathcal{L}_E(X_i) = \sum_{k\in\mathcal{K}}\sum_{\vec{q}\in X_i}(1-\mathbf{m}_k(\vec{q}))\log(\mathbf{S}(z_i(\vec{q})_k)) \quad (6)$$

3.1.2. Model metric

The model performances are compared based on the inversely weighted dice metric, defined as:

$$Dice_{wi} = \frac{\sum_{i=0}^{k-1}\left[Dice_i\left(\sum_{j=0}^{k-1} a_{i,j}\right)^{-1}\right]}{\sum_{i=0}^{k-1}\left(\sum_{j=0}^{k-1} a_{i,j}\right)^{-1}} \quad (7)$$

where $a_{i,j}$ indicates the count of observations with true class i, classified as class j.

3.1.3. Model result combination

Combining the results of the three models trained on the three geometric axes (transverse, frontal & sagittal) is a pseudo-mask of higher quality than the results of the individual models. After morphological smoothing, the pseudo mask is used to train the final model (on sagittal slices).

4. Results

The reference model was found to have a performance of $dice_{wi} = 0,76$, evaluated on the test set. This model is trained on the same dataset as the weakly supervised model, but with full label masks. Thus, this is a reference that could be described as a *classic* approach. This model is trained based on the cross entropy loss.

The first step in producing the pseudo masks is the construction of 3 models, each trained on a different set of volume slices. Each volume can be sliced to obtain a stack of transverse, coronal or sagittal slices. The performance of each model is low, compared to the reference model. Yet, combining these models allows to obtain a pseudo mask. This pseudo mask is already a segmentation mask wich shows to have a higher performance than the segmentation masks of the combined models. Since the transverse slices do not

provide sufficent context to identify the individual lumbar vertebrae, this model only segments the vertebrae semantically. It does not distinguish between $L1$ to $L5$. For this reason, the numerically higher $dice_{wi}$ score in the table is misleading.

Separate models		Combined model
Transverse	0.51	3*0.48
Coronal	0.41	
Sagittal	0.34	

Finally, this pseudo mask is used to train the resulting model. This last model has the exact same architecture as the reference model, therefor the inferrence time is identical (about 12s per volume). Its performance is an improvement of the pseudo mask performance. A performance of $dice_{wi} = 0.552$ could be obtained.

5. Conclusion

The work in (**?**) is extended with two new loss functions and a combination algorithm, allowing the model to approximate the performance of a fully supervised model at 12% of the labelling cost.

Part I

Problem Introduction & Motivation

Abstract

The objective of this project is the development of a model for the automated vertebrae instance segmentation of MRI and CT scans of the lumbar part of the human spine based on point annotated training data.

This part aims to clarify these terms. It starts with some information regarding the human spine and the medical imaging techniques to investigate it. Secondly, this part contains definitions and clarification of different machine learning problems and techniques when working with two and three-dimensional images. This part of the book ends with a discussion of previous work, both regarding the problem of instance segmentation of the human spine from medical images and previous results in the field of Weakly supervised Machine vision.

Finally, this part touches upon a possible practical application of automated lumbar vertebrae segmentation.

— 1 —

The human spine

This document presents a model for automatic segmentation of Computer Tomography (CT) and Magnetic Resonance Imaging (MRI) images of the lumbar spine. This chapter introduces these terms[1]. Secondly, it gives a basic overview of the medical imaging techniques. Finally, it touches upon a specific medical procedure in which this information is used: the minimally invasive surgery of a spinal hernia.

1. This work is not a medical desideration. For in-depth knowledge on the anatomy and physiology of the spine, consult the specialized literature.

1.1 Anatomy of the human spine

The spinal column, vertebral column or backbone [2] is a structure of 34 bones. It holds the body upright while providing it with the mobility to bend and twist. the *intervertebral discs* make this possible. These consist of a ring of fibrocartilage and an inner gel-like centre and form an articulation between two vertebrae. Moreover, the vertebral column serves as a conduit for significant nerves running from the brain to the toes. The spinal column, as illustrated in figure 1.1 can be divided into five regions:

2. NL: *wervelkolom*

the Cervical spine: 7 vertebrae of the neck, indicated by C_1 to C_7[3].

the Thoracic spine: 12 vertebrae of the middle back (T_1 to T_{12}).

the Lumbar spine: 5 vertebrae that form the lower back. These are commonly referenced as L_1 to L_5.

the Sacrum: [4] This is a structure consisting of 5 naturally fused vertebrae (S_1-S_5).

the Coccyx: [5] Structure of 3 to 5 naturally fused vertebrae at the end of the spinal column.

3. Vertebrae are numbered from the head down. C1 (*atlas*) is the vertebra closest to the head

4. NL: *Heiligbeen*

5. NL: *Stuit of staartbeen*

1.2 Pathologies of the human spine

This document does not aim to provide an exhaustive list of all human spinal pathologies. Two pathologies are interesting to discuss further as an introduction to this work since they occur in the data used in this project, see page 45 for further details.

First, there is *scoliosis*, which is a sideways curve of the spine. The severity of this condition can vary from relatively mild to severe. In severe cases, scoliosis can affect the patient's movement and breathing.

Second, there is the *spinal hernia* or spinal disc herniation[6]. A spinal hernia is caused by damage to the *annulus fibrosus*[7]. This damage can cause

6. Spinal disc herniation is sometimes referred to as a *slipped disc*.

7. The fibrocartilage ring around the softer gel-like centre of the intervertebral disc.

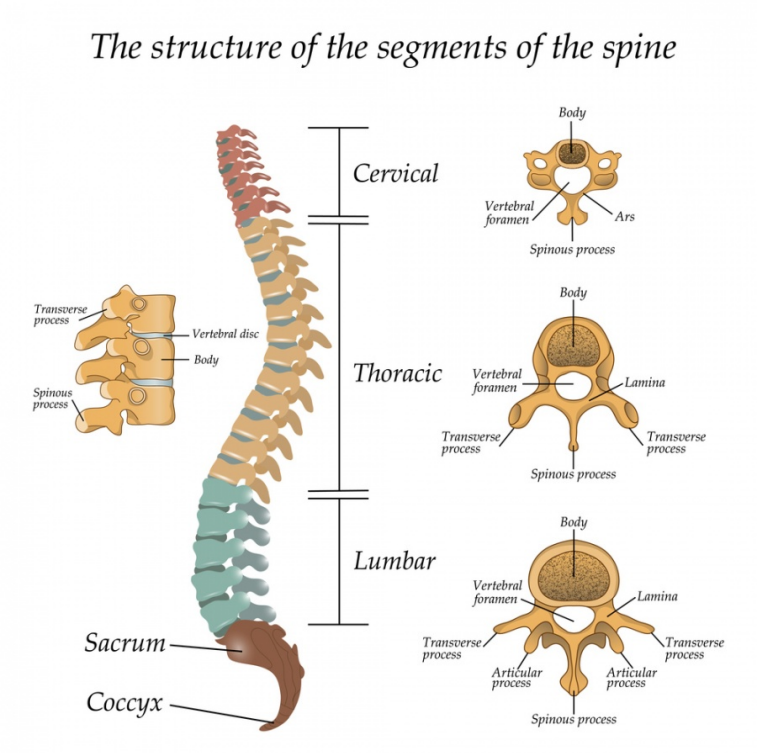


Figure 1.1: Model of the human spine. The five vertebrae in green form the lumbar spine. They are referred to as *L1* to *L5* from top to bottom. Photo source: `shutterstock.com`

Figure 1.2: Artist's impression of the start of the surgical treatment of a lumbar hernia. This image shows the dilator through which the surgical instruments will be inserted to mill away the bulging disc material. This procedure requires repeated visualization of the instruments and the spine via X-ray images. *This illustration is designed by Verhaert NP&S.*

the intervertebral disc to bulge out. This condition is painful due to the inflammation reaction, and in severe cases, the bulging material can irritate or cause impingement of the critical nerves along the spine. The nerve impingement can even lead to radiating pain to the limbs or even limb paralysis. In severe cases, a spinal hernia requires surgical treatment. This specialized procedure requires repeated medical imaging to allow the surgeon to accurately investigate the situation before the operation and assure correct positioning of the instruments during the intervention.

In image 1.2, the start of the surgical treatment procedure for a lumbar hernia is illustrated. One possible benefit of improved automatic interpretation of MRI and CT scans is providing support for this procedure. This delicate procedure requires interpretation and registration of information from both scan types (at different times). This is a demanding task for which automated support could be useful.

1.3 Medical imaging of the human spine

For diagnosis and visualization of spinal pathology several medical imaging techniques are used: Computer Tomography (CT), Ultra Sound imaging (US) and Magnetic Resonance Imaging (MRI). These techniques allow non-invasive visualization of the spine and discs in three dimensions. Two dimensional slices from volumetric scans (MRI and CT) are illustrated in 1.3.

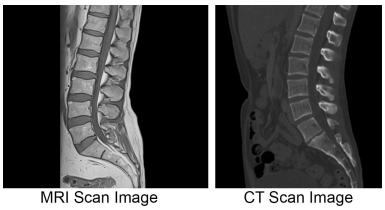


Figure 1.3: Illustration of CT and MRI images of a human lumbar spine.

1.3.1 CT scan

The Computer Tomography (CT)[8] is a non-invasive medical imaging procedure [9] for diagnostic purposes. A Computer Tomography procedure consists of the combination of an array of X-ray attenuation images taken with a rotating X-ray tube as illustrated in figure 1.4. These images can be combined with a tomography algorithm to reconstruct a volumetric representation of the radiographic density.

CT is a versatile technique with various medical diagnostic applications. Contrary to Magnetic Resonance Imaging (MRI) imaging, this technique is suitable for patients with a pacemaker or insulin pump since there are no magnetic fields involved. The main disadvantage of CT is the exposure to ionizing radiation of the patient and the risk of exposure of the medical professional to the same radiation. The image quality increases with radiation dose, but so does the probability of radiation-induced cancer. Improving the reconstruction algorithms to obtain higher-resolution images without reducing the radiation dose is an ongoing area of research.

8. Also known as CAT-scan, for Computed Axial Tomography or Computer-assisted Tomography.

9. Computated Tomography scans are primarily used for medical purposes. It is also used in industry for non-destructive component and assembly inspection. In geology, it is used to identify materials in a drill core quickly. In archaeology, it is used for the non-destructive investigation of artefacts.

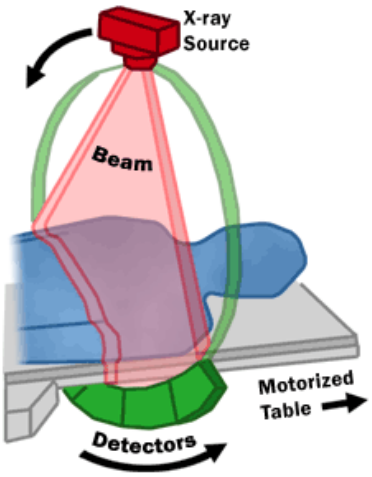


Figure 1.4: Conceptual illustration of the working of a Computer Tomography scan. Image from `www.fda.gov/radiation-emitting-products/medical-x-ray-imaging/computed-tomography-ct`

1.3.2 MRI scan

The Magnetic Resonance Imaging (MRI) scan is a medical imaging technique that is not based on ionizing radiation[10]. MRI imaging will visualize the concentration of hydrogen[11] atoms. The patient is positioned in a tunnel where a high (up to several Tesla) constant magnetic field is applied. A temporary oscillating signal is applied with the resonance frequency corresponding to hydrogen atoms is superposed on the static magnetic field. The hydrogen atoms will fall back to the equilibrium state, emitting radiofrequency (RF) signals. These can be measured by receiving coils. After excitation, the hydrogen in the water atoms in the patient's body tissues returns to equilibrium. Magnetic Resonance Imaging is particularly suitable for visualization of tissue with higher water content, such as tumours and infections, and to visualize fat.

An MRI scan can be *T1 weighted* or *T2 weighted*. The difference between these two techniques lies in whether the image is constructed based on the relaxation time of the magnetization is colinear with the direction of the static field (T1) or the magnetization perpendicular to the static field (T2). While areas with higher water content, such as infected areas, will release a higher signal on a T2-weighted MRI scan, the same images will show a lower signal strength for T1-weighted scans. The use of these different techniques of MRI scans is application-dependent.

Contrary to Computer Tomography (CT) scans, the Magnetic Resonance Imaging procedure does not expose the patient of radiation. Due to the high magnetic fields, the technology cannot be used for patients with pacemakers, cochlear implants or other metallic objects in the body. MRI allows to visualize soft tissue better than CT images. An MRI image allows visualizing both the grey and white brain matter, while this is not possible with CT images. Although both techniques produce images that resemble each other, none of both techniques can replace the other one completely.

10. Eventhough the alternative name *nuclear* magnetic resonance (NMR) might confuse.

11. In theory, other atoms than hydrogen can be excited by adapting the excitation frequency. This is rarely done.

— 2 —

Artificial intelligence & Machine Learning

Machine learning is a branch of Artificial Intelligence, The study of using computers to automatically perform tasks that once were considered only humans could do. This includes, but is not restricted to, the interpretation of speech and images. The field of Machine vision is a branch of machine learning. Some impressive steps forward have been made in the past decade, mainly thanks to the emergence of Deep Learning.

Artificial Intelligence is showing promising results for various tasks in society. New research results ranging from self-driving cars to automated translation is regularly published in mainstream media. The aim of this chapter is not to provide a complete overview of the machine learning field. Instead, the objective is to highlight concepts meaningful for this work.[1] This work investigates the use of Weakly supervised data for the training and segmentation model. The concept and benefits of Weakly supervised machine learning is discussed.

1. **Remark:** this work does not cover the social, legal and ethical questions Artificial Intelligence evokes.

2.1 Artificial Intelligence

The field of Artificial Intelligence (AI) is the engineering discipline of the automation of *cognitive* tasks. Tasks such as search, control and classification are generally considered to require a level of intelligence. Automation of this type of tasks to allow a machine to perform them is thus Artificial Intelligence[2]. A classic PID controller and even a thermostat controller can be viewed as simple but effective forms of AI. This engineering discipline has advanced considerably in the past decades, driven by leaps forward in the available hardware and new algorithms and models.

2. To be precise, it is not the human intelligence that is replicated. It is the *effect* of this intelligence.

First, the availability and reliability of hardware components such as sensors, cameras, digital storage and calculation power have increased exponentially[3]. The size and price of these components decreased equally dramatically.

3. *Moore's law* states that the number of transistors on an integrated circuit doubles every two years. This rate of progress has held more or less for a wide range of digital components in the past decades.

Second, progress has been made in developing algorithms to use this available data and computation power to solve problems and perform tasks. This text does not provide a complete overview of all existing machine learning models. This work makes use of Deep Learning models. A Deep Learning model is a type of Artificial Neural Network (ANN) with multiple hidden layers. Deep learning models are behind almost all recent applications of Natural Language Processing (NLP) and Machine vision.

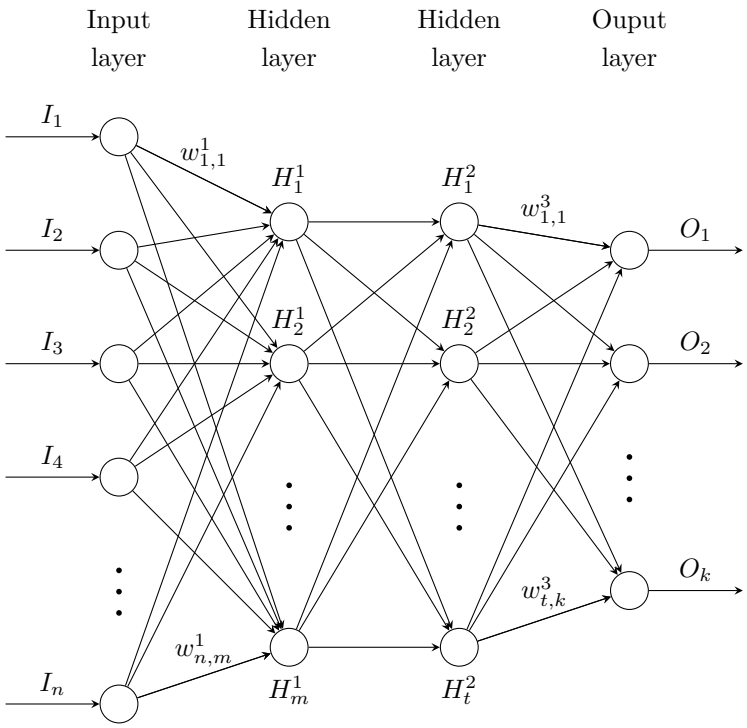


Figure 2.1: Concept of an Artificial (feed-forward) Neural Network. The illustrated network has 4 layers: the input layer, two hidden layers and the output layer. The hidden layers first perform a feature extraction transformation on the input data. Based on these features, the output layer generates the model result.

2.1.1 Neural Networks

General neural network architecture

An ANN is a collection of connected nodes, or *neurons*. These are typically structured in layers, such as in figure 2.1. There is always an *input* layer and an *output* layer. Between these two, one can find the *hidden* layers[4]. To calculate a node value, the incoming node values are weighted by the connection weights and the result of this linear combination is transformed by an activation function[5]:

$$z_j(\vec{x}|\vec{w}) = \frac{\vec{w}^T\vec{x}}{\sum w_k} \tag{2.1}$$

$$y_j(\vec{x}|\vec{w}) = \sigma(z_j) \tag{2.2}$$

The simple network in figure 2.1 illustrates different characteristics of neural networks: First, this concept is flexible. Architectures with more than two hidden layers are common. The number of neurons in the hidden layers is another degree of flexibility. Second, a neural network has a high number of parameters. This small network consists of $n \times m + m \times t + t \times k$ weights w and the activation functions in each layer. The concept of a Convolutional Neural Network (CNN) is to reduce the number of weights in specific layers by imposing these to represent a single convolution filter.

The parameters defining the architecture of the network, such as the number of hidden layers, the number of nodes per layer and the design of possible convolution layers, are called the model *hyperparameters*.

Convolution neural network

ANN models can be trained to perform numerous different tasks due to the high number of degrees of freedom in the model[6]. However, this high number of degrees of freedom also entails some problems and difficulties:

4. When there is at least one hidden layer, one talks about a deep network. In practice, CNNs have multiple hidden layers.

5. $\vec{x}$: values of the nodes connected to node j
$\vec{w}$: connection weights
$\sigma(.)$: activation function (e.g. ReLu, tanh, ...)

6. An in-depth discussion of neural network theory is outside of the scope of this book. This text aims to reiterate critical concepts the reader is assumed to have some prior familiarity with. There are several valuable sources available for readers who are not satisfied with the level of detail provided here. Books such as `Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow` by Aurélien Géron or `Deep Learning`

- The model weights have to be stored in computer memory. This is limited and restricts how deep[7] the network can be. A deeper network can be beneficial because it can extract more high-level Features by combining the lower level features in the preceding layers.
- An over-parametrized network is very prone to overfitting on the train set. An overfitted model has not learned the generalizing relationship between the input and the labels but rather the quirks of the specific train set used.
- There are some more technical problems, such as *vanishing gradients* that are associated with naively copying the network shown in figure 2.1.

7. The depth of a network is the number of hidden layers.

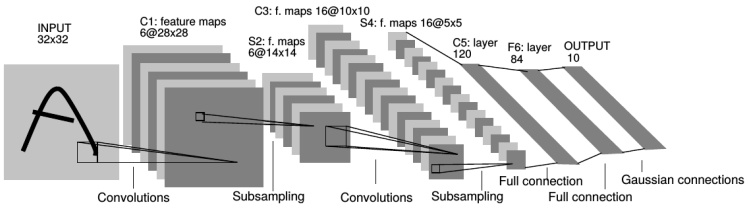


Figure 2.2: Illustration of LeNet, a famous Convolutional Neural Network (CNN) architecture. Image is taken from the famous paper `Gradient-Based Learning Applied to Document Recognition` by Yann LeCun, Léon Battou, Yoshua Bengio and Patrick Haffner.

In search of solutions to these difficulties, Y. LeCun et al. proved the benefit the CNN concept (figure 2.2). The convolution layer is suitable for data with a grid-like structure, such as images or volumes. The structure of the data allows a high level of weight-sharing. Neurons in a convolution layer are not connected to all nodes of the previous layer. Instead, each has a limited receptive field. Regardless of the size of the image, a convolution layer with a 5×5 convolution filter *kernel* only has 25 weights per channel. If the input of the layer is an RGB image with three channels, this sums to 75 parameters. A 5×5 filter is larger than usual. The number of pixels processed by the filter is called the *kernel size*. Typical kernel sizes are 2×2 pixels or 3×3 pixels. One can choose to ignore pixels within a kernel. A dilution of 1 on a 3×3 kernel expands this kernel to cover a 5×5 patch, processing 9 evenly spaced pixel values.

In figure 2.2 this idea is illustrated. The nodes of the feature maps in stack $C1$ are all constructed based on a small filter kernel shaped patch of the input image. By systematically applying the same filter to each overlapping filter size patch of the input a feature map is created. Figure 2.3 illustrates the procedure. Each node of the feature map is the scalar product[8] of an image patch and the convolution filter followed by the activation function. A filter can detect a specific feature in the input, a horizontal line for example. Convolution layer $C1$ has different filters. It will produce different feature maps. This means that 6 different basic features can be extracted by this layer, e.g. horizontal lines, vertical lines and right or left slanted lines[9]. By stacking different convolution layers, the network can learn higher-level features by combining the lower-level features. Combination of low level features such as lines can allow the network to detect higher-level features such as shapes (circle, rectangle, ...). These higher level features can be combined to detect even higher-level features such as a house or a face.

8. The scalar product or dot product is the element-wise product of the arrays followed by the sum of the elements. The result is a scalar.

9. These filter kernels are trained on the train data by the backpropagation algorithm. The mentioned features can result from this training procedure but will not be imposed by the engineer.

A convolution layer in a CNN can train a high number of filters in parallel. It is common a convolution layer learns 32, or even 512, filters in parallel, extracting as many different features from the input.

Hyperparameters defining the size of the output of a convolution layer are the depth, the stride and the padding size. Figure 2.3 illustrates how the filter is applied to overlapping regions of the input. When the stride is 1, the filter is moved a single pixel. Larger stride values reduce the dimensions of the output and the overlap because the filter is translated further. In some cases, it is convenient to pad the input with zeros (or other values) to give further control of the output size.

Often, a convolution layer is followed by a *pooling* layer as a dimension reduction step. Pooling layers do not contain learnable parameters.

Many segmentation networks consist of en encoder and a decoder part[10]. The encoder part consists of a sequence of convolution layers and pooling layers. This part of the network calculates the image *encodings* deep feature maps with dimensions lower than the input image dimensions. The second part of the network is the decoder. This part of the network will decode the encodings to a segmentation mask with the same dimensions as the input image. Simple techniques for upsampling such as *nearest-neighbour* or *interpolation* are sometimes used. A more interesting technique is the *transposed convolution*[11]. This technique is very similar to a regular convolution operator. It is based on a filter with trainable weights *sliding* over the input. Output node values are again calculated as the scalar product of the input and the filter values. Below, one can see how some convolution parameters are interpreted just the other way around for this operation. Where a convolution layer with stride two will result in an output with lower dimensions than the input, a transverse convolution will expand the input. To calculate a transverse convolution with stride > 1, one will add empty cells between the input values. One could say the filter steps are in this case $\frac{1}{2}$ cell. This operation results in an output feature map with higher dimensions than the input feature map. In the case below, the 2×2 input is upsampled to a 5×5 feature map.

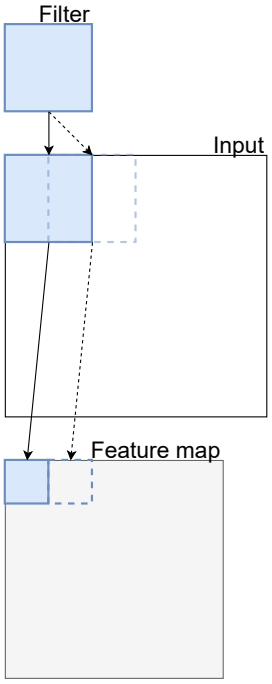


Figure 2.3: Repeated filter application to produce feature map. Image inspired by `www.machinelearningmastery.com`

10. examples of such architectures are the VGG16-FCN8 or the UNet architecture, illlustrated on page 35.

11. Sometimes, the transposed convolution is referred to as the *deconvolution*. This is not an exact formulation. It is not intended to reverse the effect of any convolution operator.

Input		Input (add zeros)							kernel			output				
		0	0	0	0	0	0	0								
		0	0	0	0	0	0	0				3	6	12	6	9
3	3	0	0	**3**	0	**3**	0	0	2	1	2	0	3	0	3	0
1	1	0	0	0	0	0	0	0	0	1	0	7	5	16	5	9
		0	0	**1**	0	**1**	0	0	3	2	1	0	1	0	1	0
		0	0	0	0	0	0	0				2	1	4	1	2
		0	0	0	0	0	0	0								

Table 2.1: Transposed convolution to upsample a 2×2 input image with a 3×3 kernel and stride 2. The indicated value of the output feature map is calculated as the scalar product of the indicated part of the input and the kernel. Example inspired by `https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8`.

Training a neural network

Constructing a network requires evaluating it, comparing the evaluation output to a known desired output, and taking steps to bring the model output closer to the desired output. The procedure used to fit the weights of a neural network to the training data is the optimization algorithm. A network is

fitted to the *train set* by the optimization algorithm, based on the **loss**. This optimization algorithm for a neural network is based on the gradient descent algorithm. By iterated step in the opposite direction of the gradient, this algorithm is capable to approximate a (local) minimum of a differentiable function. Thus, the model loss is required to be a differentiable function. To minimize the loss function F, the network parameters θ are updated in the direction of the negative gradient $-\nabla F(\theta_n)$:

$$\theta_{n+1} = \theta_n - \gamma \nabla F(\theta_n) \tag{2.3}$$

To optimize the weights of the hidden layers, the gradients have to be *propagated back* through the network layers. This operation can be iterated until the solution converges. Often, the procedure is stopped earlier. The ML engineer wants to judge the performance of the model based on one or several **metrics**, calculated on the *train set*, the *cross-validation set* and eventually on the *test set*.

The evaluation on the test set, separated from the train set on which the model is fitted, is done to provide the best possible estimation of the model performance on unseen data. When training a model, the objective is to obtain a model which generalizes well to future problems[12]. One wants the model to learn the general relationship between the input data and the output. A model which is capable of producing the outputs for the specific set it is trained on but is not capable of doing this for new data is *overtrained*. It has learned the specific quirks of the train set rather than the generalizing relationship[13]. To avoid this, or at least identify the problem, the evaluation on the cross-validation set is necessary. To avoid overfitting, the model, while training with gradient descent optimization (equation 2.3), is evaluated on the cross-validation set. Once the training of the model on the train set is not improving the metric performance on the cross-validation set, the model is being overtrained. Further reducing the loss does not result in a better generalizing model.

12. Provided these future problems are samples from the same population as the train set.

13. It is also possible that the new data is sampled from a different population than the train data. In this case, the model might have learned the desired relationship for the original population, but this relationship turns out to be different for the population in practice.

The gradient descent step in equation 2.3 can be evaluated after every individual sample. For the machine vision problem described in this work, that would mean after every image (crop) of the train set. This approach is called *stochastic* gradient descent. The model weights are updated very often, but possibly not always in the most optimal direction. One can also choose to update the model weights only once all the images in the train set have been evaluated (one *epoch*), accumulating the gradient direction. In this case, there is less stochastic variation on the gradient direction, but the network weights are updated very slowly. This work uses *batch* gradient descent, updating the model weights every 6-image batch[14].

14. The 6-image batch size is based on the restriction on internal memory of the used GPU

2.1.2 Artificial intelligence for healthcare applications

Various researchers are exploring the opportunities of Artificial Intelligence in medical practice. This could reduce the burden of repetitive tasks on medical caregivers and support both medical diagnosis and procedures.

Artificial Intelligence applications can support medical professionals by allowing more cost-effective solutions to monitors patient's condition. One example is the use of a Photoplethysmogram (PPG) signals to estimate a patient's blood pressure. Blood pressure is a valuable indicator of the patient's condition for the medical staff. Measuring blood pressure is time-consuming

and disturbing for the patient. However, the PPG signal is relatively easy to measure. This only requires a watch-size sensor[15] around the wrist, that can be worn continuously. Inferring the blood pressure from such a PPG signal[15] allows the healthcare worker to access valuable information continuously, cheaper and without much inconvenience for the patient.

Different researchers investigate Machine vision applications for medical applications. Research on image classification for medical diagnosis is conducted among others for melanoma (skin cancer) diagnosis on camera (RGB) images [31].

15. Several commercial sports watches already use a PPG sensor to measure athletes hear rate.

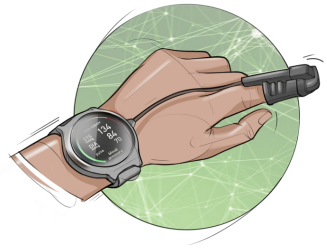


Figure 2.4: Illustration of a ppg sensor, typically worn around the wrist. Apart from the heart rate measurement, this signal can be used to estimate the patients blood pressure without much effort from the medical staff or inconvenience for the patient. Image provided by *Verhaert NP&S*

2.2 Machine vision

Machine vision is the branch of Artificial Intelligence focussed on image processing. The machine vision task performed in this work is called (instance) Segmentation. This chapter explains what this means. The segmentation task is compared to other machine vision tasks.

2.2.1 Machine vision tasks

Machine vision is a broad discipline. Humans extract information from images almost subconsciously, and we are often not aware of the different tasks we perform on images. The objective of this section is to briefly define different machine vision tasks discussed further in this book. Several machine vision tasks consist of *recognizing* objects, animals or humans in an image. A model is built for a finite list of *categories* that can be present in an image. Depending on the question asked ad inference time, one can distinguish the following tasks, which are also illustrated in figure 2.5.

Image classification is the task of determining what object category[16] is present in the image, e.g. *Is there a sheep in this image?*

16. A slightly more advanced task is to be able to distinguish several classes in one image.

Object counting is the task of counting how many instances of each category are in the image, e.g. *How many sheep are there in this picture?*

Object detection consists not only of identification of the object. Also, the spatial position is requested, for example, in the form of a bounding box. *Where is the sheep in this picture if a sheep is present?*

Semantic segmentation requires a class estimation for each image pixel. Pixels that do not belong to a specific class are called the *background*.

Instance segmentation requires not only that the semantic class is determined for each pixel, but also that two individuals of the same class[17] are distinguished.

17. For each pixel, the model predicts not only if it belongs to a sheep, but also which of several possible sheep it belongs to.

The different machine vision tasks above are ranked in order of how informative the model output is. It is clear that only saying if a cat is present in an image is less informative than indicating pixel-per-pixel where this cat is in the image.

2.2.2 Data for training machine vision models

To perform the tasks discussed in chapter 2.2.1, one needs to build a suitable model. For Machine vision tasks[18], the current standard approach is Deep

18. and many other tasks.

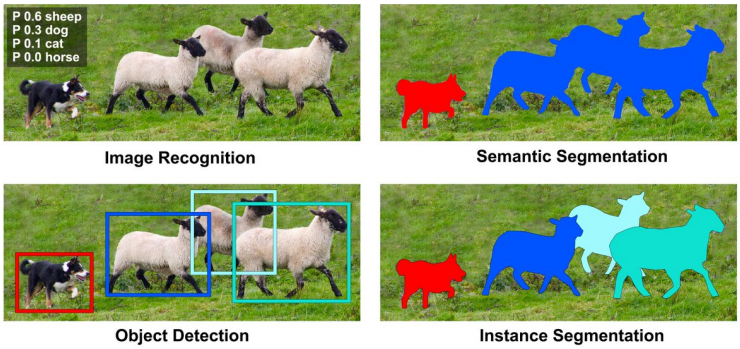


Figure 2.5: Illustration to compare different Machine vision tasks [6]. Object detection means that the location of several objects is estimated by the model. This is indicated by the *bounding boxes*. Segmentation of an image is classifying each pixel in the correct class or assigning it to the *background* class. Semantic segmentation makes no difference between different instances of the same semantic class, instance segmentation does.

Learning. The cost to generate, store and communicate images and computation power has dropped in the past decades. This evolution allows to train a model on previously unimaginable quantities of data[19]. This technique allows a model with a high number of degrees of freedom to be trainded without the need for expert-crafted Features.

19. The *ImageNet* database (`http://image-net.org/challenges/LSVRC/index`) consists of more than 14.10^6 images.

Weak supervision types

To build a model to perform the tasks discussed in 2.2.1, this model needs to be trained. This requires a set of *labelled* images. Collection of this dataset - and the dataset labels - proves to be a challenge. Training a model with *weak labels* compared to *strong labels* could help to mitigate the labelling cost problem.

In the classic approach, the supervision type closely resembles the intended model output, the Ground truth. To train a model that can classify an image[20], one has to *train* the model on a set of labelled images where a human indicated the class. To train a model to perform image segmentation[21], an expert needs to provide a set of images where for each picture, the objects are delineated.

20. Given an image, the model outputs if this picture is a representation of class *cat*, *dog* or another animal or object.

21. segmentation means that the model classifies each pixel.

The idea behind Weakly supervised is to train a model with cheaper annotations that contain less information than the desired model output, making use of the latest information available in the labels without explicit indication. Figure 2.6 illustrates several types of image supervision : From left to right on the top row, this shows point supervision and squiggle annotation, common annotation types for weak supervision. On the second row, bounding box annotation and complete mask annotation are illustrated.

The objective of Weakly supervised is to construct a robust model based on *cheap* (incomplete, noisy or imprecise) labels, sometimes described as *indirect supervision*. Numerous creative approaches have been conceived. Since the provided annotations in Weakly supervised are not full labels, these are sometimes described as *hints* instead [4]. The basic concept of Weakly supervised is that there are two sources of information to draw from: The hints and the prior knowledge about the problem (Priors). These *Priors* can be any form of prior knowledge about the object to be segmented[22]. Priors can be the object size, shape or location, the number of instances, the similarity across images or the similarity with external images.

22. or any other machine vision task.

Whether an annotation is considered a *weak label* or a *strong label* de-

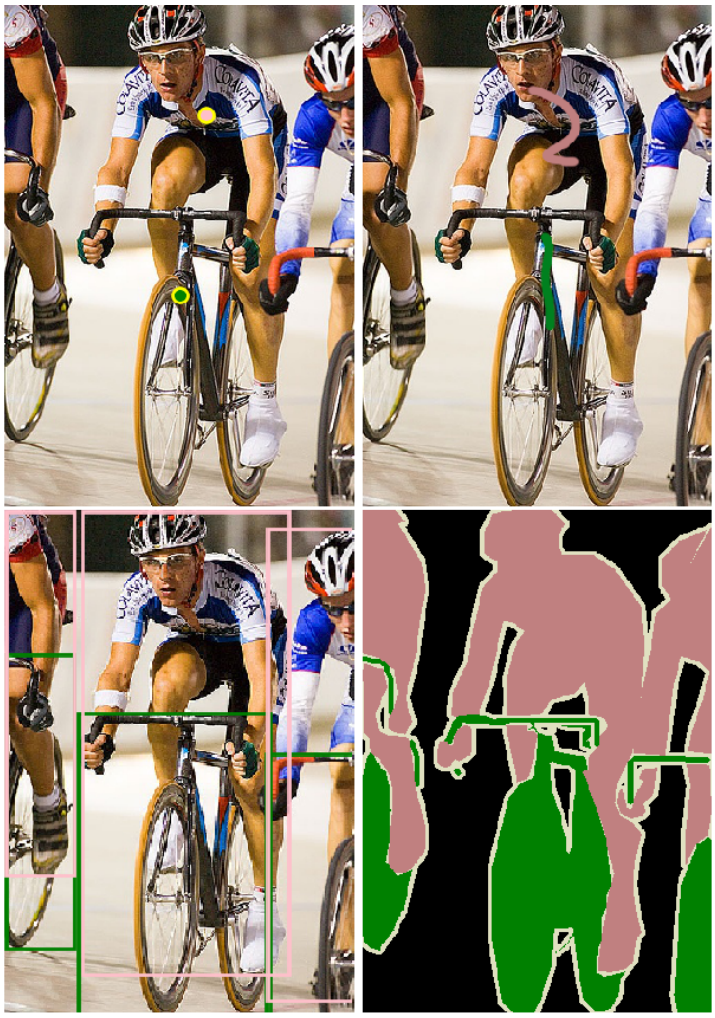


Figure 2.6: Four different annotation types [26]: On the top left the picture is point level annotated. The points are inflated for visibility. On the top right, squiggle annotation is used. The bottom left shows bounding box supervion. While the bottom right image is fully annotated. An image level label would indicate that there are multiple instances of *person* and *bike* in the image.

pends more on the modeller's intention than on the annotation itself. When one aims to construct a model to infer output labels with a higher informative value than the original annotations, these *labels* become *hints*. Making a model predict bounding boxes from a dataset annotated with bounding boxes means considering these as *strong labels*. If one uses the same dataset to construct a model that predicts pixel-wise masks, the labels are *weak labels* or *hints*. For a segmentation task, weak labels can be:

Image level labels : In this case, only the object's class in the image is provided. This would be a full label for a classification task, but it is a weak label for a segmentation task[23].

Point annotation : This annotation technique, the subject of this work, consists of asking the expert to indicate the classes with one or several points. This technique is used in [20, 21, 26]. In [25], instead of *random* point annotation, the author makes use of *extreme* point annotation. This work chooses to investigate random point annotation. This is believed to approximate the fastest and most natural action of simply pointing at an object best.

Squiggles : This annotation technique is related to the point annotation technique. Instead of points, the expert is asked to indicate the classes with a squiggly line.

23. Image with image-level annotation can, for some problems, be obtained in large numbers by web-scraping[30].

Bounding boxes : A bounding box (a rectangle circumscribing the object) is a less precise form of object localization than pixel-wise segmentation.

Image description : This task combines the problem of Natural Language Processing with the problem of image segmentation. The annotation of the image is derived from a verbal or written description of the image. This annotation type has not yet been used by many researchers. It might be promising since large bodies of datasets could be available from, for example, medical files where a medical expert has provided a written diagnosis based on available medical images.

There seem to be endless variations possible to conceive weak labels. For example, instead of image-level labels, in [21], the number of instances of the specific class[24] is provided.

24. Which happens to be penguins.

Bearman et al.[3] compare the annotation time required for images from the PASCAL VOC 2012 dataset. The authors report an average time per image of 239.7 seconds per image for full image pixel annotation, 20.0 seconds per image for image-level labels and 22.1 seconds per image for point annotation. As is repeated in 3.2, point level annotation is attractive since it barely requires more of the expert's time while delivering more precise information. The point label contains valuable localization information, absent from the image level annotation. Besides this, the reported cost for labelling additional instances of the same class is only 0.9s per addition instance. The image labelling time consists of a *startup cost* of 18,5s followed by the time to annotate the first instance of the first labelled class of 2.4s. This means that the time to label the 2,8 class instances that are on average present in an image from the PASCAL VOC 2012 dataset takes 23,3 s per image. The observation that labelling additional instances of a class results in such a low increase in labelling time is used in this work. In chapter 4 on page 62, the time to provide multiple labels for the *same* class instance will be estimated based on the same addition of 0,9 s.

Techniques to train models on weakly supervised data are often based on the construction of a *pseudo* mask. This pseudo-mask is then used as a label to train the network. This involves an extra step in the training process. These techniques are more computationally intensive than those on fully Supervised data, since the construction of the pseudo mask involves an additional computational step. Apart from this, the loss function can consist of multiple parts and might involve repeated evaluation of the model on the same batch[25]. Weakly supervised learning is based on the idea that the cost of human experts is fixed or even increasing while the cost of computation power is decreasing. Besides this, the ability to label more data at the same labelling price point can increase the variability of the data on which the model is trained. This can increase the robustness of the final model obtained.

25. The consistency loss used in this work requires to evaluate the model both on the input image and the rotated version of this input image.

Whether a network is trained on pseudo labels or on fully supervised labels does not change the evaluation effort for that model. Some extra steps might be required to train a model on weakly supervised data compared to training a model on full labels, but the model evaluation at inference time is therefor not more computationally intensive.

— 3 —

Previous work

This chapter gives an overview of the previous work in two fields combined in this project. First there is the application of Artificial Intelligence[1] in medical applications, specifically for Machine vision problems. Then, there is the active area of research of Weakly supervised machine learning both for medical and for non-medical applications.

1. For a brief introduction of Artificial Intelligence and Machine vision itself, please consult chapter 2.

3.1 Machine vision for medical applications

For segmentation tasks, the U-net architecture[27] is widely used[2]. This architecture can be represented by a characteristic U-shape (see figure 3.1), as the name indicates. This architecture is based on a *contracting* resolution path followed by an *expanding* path. The contracting path refers to the resolution reduction in each step. These steps consist of two convolution layers, of which the result is used in two ways. First, the convolution layer result is *down-sampled* (the layer dimension is reduced) with a max-pooling layer. In each contracting step, the number of feature channels is increased, allowing the network to propagate context information and produce feature maps with more high-level features. Second, this convolution layer result is concatenated with the corresponding feature maps in the expanding path. These expanding steps consist of inverse convolution layers, expanding the layer dimensions. To some extent, the expanding path is symmetric to the contracting path, giving the network its U-shape. The connections where the convolution layer results of the contracting path are concatenated to the corresponding layers in the expanding path are called the *skip connections*. Through these connections, the network can propagate information efficiently over different resolution levels[3]. The 2-dimensional network developed in [27] was first adapted for three-dimensional problems in [9].

2. The U-net architecture was developed specifically for biomedical applications.

3. Skip connections have proven efficient tools to decrease problems such as gradient vanishing.

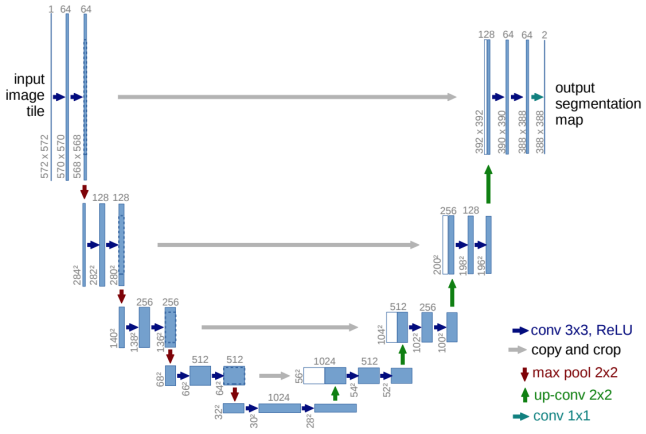


Figure 3.1: U-Net architecture, as illustrated in [27]. Each blue box represents a multi-channel feature-map. The number of channels is indicated above the box, the $x \times y$ dimensions are indicated at the bottom left. The gray arrows indicate the feature maps in the contracting path are copied and concatenated to the feature maps of the expanding path.

Various segmentation architectures are based on similar ideas: first, a high-level feature map is created in a contracting path. This feature maps is sometimes called the image *encoding*. Subsequently, this feature map is *decoded* in an expanding path to form the segmentation mask.

3.1.1 Automated segmentation of the human spine

Multiple researchers have build systems for the automated segmentation of the human spine. Historically, this was done with techniques based on the calculation of mathematical and geometrical features of vertebrae. For example in [16], a model registration approach is used to segment a set of 10 CT images. A parametric geometrical model of a spine is optimized to fit the scan geometry as good as possible. These techniques showed to suffer from difficulties to generalize [4]. When the scan geometry does not match closely to an ideal spine, for example, when a patient has severe scoliosis, this approach runs into problems. Before the general introduction of deep learning networks, techniques such as Viola-Jones detectors were used, such as in [34]. A project of which the dataset is discussed on page 50.

Deep learning models were developed for the problem [28, 14, 8, 24]. The model in [28][5] uses a multi-step approach. First, the Region of Interest (RoI) (the lumbar region) is selected using a multilayer perceptron evaluated on the response of a Canny Edge filter. This RoI is then sliced (sagittal) and crops ($270mm \times 270mm$[6]) are segmented with a 2D U-net. The resulting masks are improved with a morphological closing filter. In [14], a similar 2 step approach is used. Here, the author chose to use a 3D U-net to evaluate patches from the RoI instead of a 2D U-net.

An interesting approach to the problem is published in [24, 8] by Lessman and Chuang. The approach presented in this work makes use of the available prior anatomical knowledge of the human spine: the vertebrae are located right next to each other and always occur in the same, known order. The basis of the network in [24] is an extended 3D U-Net, combined with an elaborate inference scheme. In [8], this network was improved to reduce the memory required and the complexity of the inference scheme.

In figure 3.2, the inference scheme used in [24] is illustrated. This approach is based on the anatomical knowledge that vertebrae occur in order, right next to each other. The network, illustrated in figure 3.3 in the version as presented in [8], performs instance segmentation of the scan by sequentially evaluating the model on 3-dimensional patches of the scan. These patches have dimensions $180mm \times 180mm \times 180mm$ and can thus contain a complete vertebra and meaningful parts of one or two adjacent vertebrae. The model is trained based on an image patch combined with a memory patch. This memory patch contains the voxels of the vertebrae that were already segmented in the sequential procedure. The network is trained to segment only the consecutive vertebra in sequence. For inference, the procedure evaluates patches from the scan border until a (partial) vertebra is discovered and segmented. After storing this first segmented vertebra in the memory patch, the next vertebra is segmented from a patch that is centred on the previous vertebra and shifted down [7]. Following this procedure, the vertebrae are segmented one by one. This procedure was developed on a collection of 103 medical images of the human spine, including the xVertSeg dataset (discussed on 48). To improve the segmentation result, a *soft false positive* and

4. Before the general introduction of deep learning, limits to the model's capability to generalize were observed in many different applications. This could be improved with deep learning networks, at the cost of requiring higher data volumes, higher data variability and often higher computation cost. Simply put, the higher number of model degrees of freedom, combined with the ability to work without human encoded features, allows a better representation of the complex and variable reality.

5. This model is developed on the xVertSeg dataset, which is one of the datasets used in this work. It is described starting on page 48.

6. The article also mentions $27mm \times 27mm$, which seems to be a typo.

7. The procedure works both for sampling up-down and down-up. Here, I stay with the procedure illustrated in figure 3.2.

soft false negative loss is added to the cross-entropy loss, where the voxels are weighted by the inverse of the distance to the segment border. These terms increase the loss specifically for those voxels close to the edge of a vertebra.

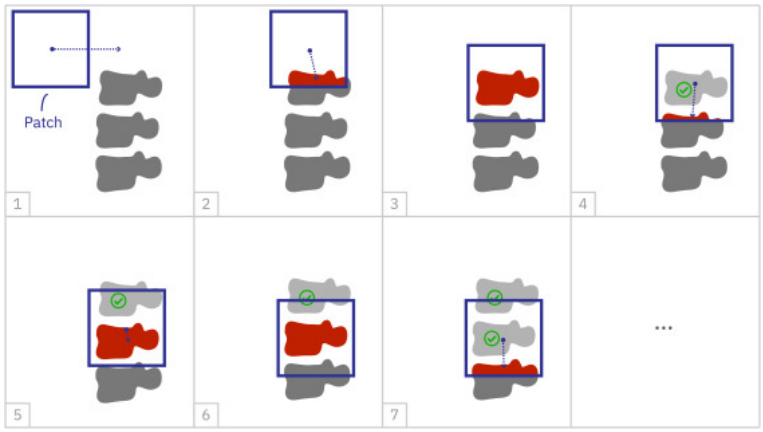


Figure 3.2: Illustration of the inference scheme used in [24] (image taken from same).

In [8], some improvements to this procedure were introduced. The model is rendered less memory-intensive bypassing the feature maps in the skip-connections after unspooling instead of directly. A method is developed to improve finding the starting point of the sequential segmentation procedure by evaluating only the patches where a high bone density is observed. The network is trained to output 3 values:

classification C : A classification result of the segmented vertebra (in the specific evaluation step).

instance segmentation mask S_1 : A segmentation mask that indicates to what vertebra instance every voxel in the patch belongs.

Semantic segmentation mask S_2 : A binary segmentation mask that indicates whether a voxel belongs to a vertebra or to the background class.

To some extent, these three outputs seem redundant. Labelling the individual vertebrae is a challenging task since there is a very high geometric correlation between vertebrae. Thus, a combination of different estimations is intended to procedure more reliable labelling.

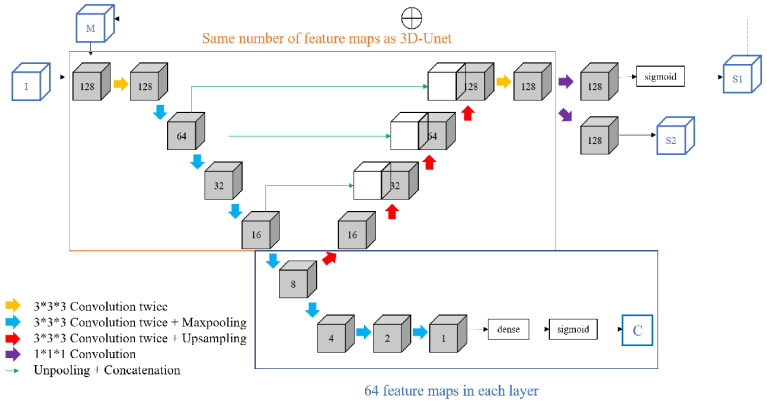


Figure 3.3: Extended U-Net architecture. Image from [8]. This network has

3.2 Weakly supervised segmentation

The subject of Weakly supervised is broad. Researchers devised various ideas to decrease the labelling cost by leveraging weak labels to infer strong results. Chapter 2.2.2 discribes a number of different annotation types.

3.2.1 General overview of techniques and approaches

To solve the weakly supervised problem, different authors[19, 18, 1, 26] share the same high-level technique to work in a two-step procedure, first estimating point guided pseudo-masks, then training a network based on these pseudo-masks. The construction of these pseudo masks is performed differently.

Given an image-level class label, the Grad-Class Activation Map (CAM) [29] allows to identify the picture regions responsible for a classification result. The basic idea is to trace back the gradient to identify pixels for which changing the pixel value would change the model classification output. The pixels that incite the model to predict a certain class correspond to images of this class. In [29] these are used as pseudo masks to train a segmentation network, only based on image-level labels. Ahn et al. [1] combines the pseudo masks generated with CAM with an inter-pixel relation map. Since the mask generated by using CAM cannot distinguish between different instances of the same semantic class, it is not suitable for generating instance (pseudo) masks. In [1], these masks are combination with (semantic class agnostic) inter-pixel relation maps that can make a difference between different instances. This inter-pixel relation map is trained by encouraging the model to identify an instance boundary between pixels with different class affinity and by identifying a limited number of instance centroids. Other pseudo mask generation methods exist, for example methods based on partial image occlusion or erasing[32]. In this approach, image regions associated with different classes are identified by training an adversarial network to erase regions of the image to change the classification output. The idea is identical to the approach based on CAM: detect the regions of the picture responsible for the classification result.

The techniques mentioned above make use of image level annotation. This work is based on weak labels containing a higher level of information: point annotation. The rationale behind point annotation is that when an expert is required to label the images, optimal use needs to be made of the expert's time. Based on estimations [3] the time consumption of point annotation and image level annotation is small, yet the information quality is higher (see also chapter 2.2.2). An expert does not need significantly more time to click on an object compared to providing an image label[8], yet the point provides valuable localization information. In [26], the case is even made that for large and dense images, point annotations could even help the expert to keep track of the already labelled objects. By using the annotation points, in a technique called Point supervised Class Activation Map (PCAM), the authors of [26] can improve the quality of the class affinity estimation compared to the regular CAM technique (using the PASCAL VOC 2012 dataset). The obtained affinity maps are subsequently used as *pseudo* label maps to train a network (ResNet50). The author reports an improved performance of the model trained on *pseudo* labels compared to the performance metric

8. It should be noted that point annotation refers to *random* point annotation. Some authors [25] use *extreme* point annotation. Where the expert is requested to indicate points at the edge of the segment region.

calculated for those pseudo labels.

3.2.2 Weakly supervised segmentation for Medical applications

dr. I. Laradji developed several methods to make use of point supervised data for segmentation tasks. First, LC-FCN is introduced: Fully Convolutional Neural Network (FCN) with a Location based Counting loss (LC) as optimization loss. This technique was introduced by I. Laradji et al. in [21] for the training of an automated instance counting network on point-supervised data. Later it is applied in [20] as the Location branch of the WISE network. Contrary to other Weakly supervised projects, the authors focus on multi-class, multi-instance problems for which CAM-based solutions are less suitable. This method, illustrated in figure 3.4, consists of two branches from the same backbone: the Embedding branch and the Localization branch. The Embedding branch is trained with a pre-trained class-agnostic object proposal method. This network (FCN8) is pre-trained to output similar embeddings for pixels that belong to the same class. The output of the network is thus an Embedding mask for each pixel of the image. This network is trained to deliver different embeddings for different object instances with a binary cross-entropy function on the squared exponential kernel difference of the embeddings [9]:

$$
\begin{aligned}
S(i,j) &= \exp\left(\frac{||E_i - E_j||^2}{2d}\right) \\
\mathcal{L}_E &= -\sum_{(i,j)\in\mathcal{I}} [\mathbf{I}(y_i = y_j) log(S(i,j)) \\
&+ \mathbf{I}(y_i \neq y_j) log(1 - S(i,j))]
\end{aligned}
$$

9. With E_i and E_j the embeddings for points i and j and $\mathcal{I}$ the collection of labelled points.

The second part of the WISE network is the Localization branch. This branch aims to predict small *blobs* at the centre of each object. These blobs are too small to serve as masks on their own, but they can be combined with the E-branch result to provide segmentation masks.

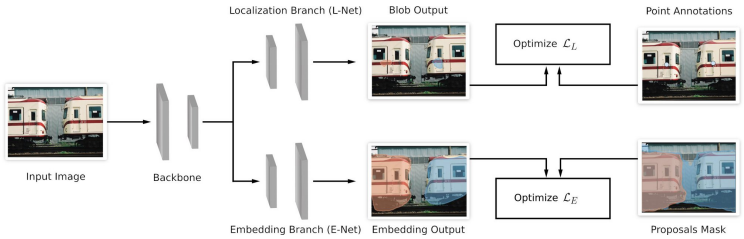


Figure 3.4: Illustration from [20]. The WISE approach consists of two branches: The Embedding branch and the Localization branch.

The WISE model seems to perform well on the PASCAL VOC 2012 dataset, the Cityscapes dataset and the COCO 2014 dataset. Nevertheless, the author abandoned it in [22, 23], where he demonstrated a weakly supervised network to segment patient lungs and zones of opacity and consolidation in the lungs of potential Corona virus disease '19 patients. There are several potential reasons for this. Two of those seem essential for this work. First, the WISE approach seems to assume the expert positions the annotation point right in the centre of each object instance. Second, the approach is complex, and the double branch inference is computation and memory intensive. This project assumes that experts can provide different point labels for the same instance at random locations in the instance. Since volume data is is more memory intensive than picture data, it is beneficial to avoid overly complex and memory-intensive approaches.

In [22], Laradji et al. introduces an approach based on the consistency loss, combined with binary cross-entropy. This approach, illustrated in figure 3.5 is used and extended in this project. Details are provided starting on page 28.

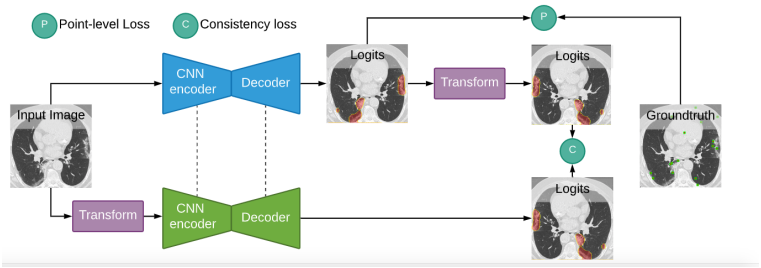


Figure 3.5: Illustration from [22]. The consistency loss approach is based on the combination of two loss terms: the Unsupervised consistency loss and the (weakly) Supervised point (cross entropy) loss.

Figure 3.5 illustrates the basic concept of the consistency model. The network output is required to be consistent under a geometric transformation, i.e. the network output when presented with image $\mathcal{X}_i$ rotated over 90° should be the rotation of the network output when presented with image $\mathcal{X}_i$.

Part II

Modelling methods & Analysis

Abstract

This part of the document details the used datasets and the modelling and evaluation methodology. First, and overview is provided of the 5 datasets used in this work. Statistics regarding the patient's age and gender are documented for the sets for which this metadata is available. The aim is to investigate possible biases in the data.

Second, the data preparation steps are explained, before describing the modelling approach, the network architectures and the model loss functions.

— 1 —

Data preprocessing

The available data volumes have to be preprocessed before being used in the neural network classifier. The preprocessing steps consist of resampling, slicing, contrast enhancement and cropping.

There is not one obvious way to combine the different datasets and available labels in one project. The chosen approach in this work is discussed below. First, the conversion of the full class labels to point labels is discussed. Secondly, the split of the datafiles in a train set, a validation set and a test set is discussed.

1.1 Image preprocessing

A (medical) image[1] consists of a combination of data - the pixel [2] values - and metadata.

For a medical image, there are two types of metadata:

Describing the patient & medical data: patient identification, pathologies, date of scan, date of birth, gender, weight, height,...

Technical meta-data: The size (number of pixels per dimension), pixel spacing (distance between pixels along a dimension), orientation vector, image modality & other information regarding the image acquisition.

For a discussion of the data used in this work, chapter 1 can be consulted.

1. In this work, I use *image* to refer both to 2-dimensional and 3-dimensional images.

2. A *pixel* (picture-element) is the smallest component of a digital bitmap image. It is defined by a position vector and carries c values. Where c is the number of channels of the image. Sometimes, a pixel of a 3-dimensional digital image is called a *voxel* (volume element).

1.1.1 Resampling and slicing

In figure 1.1, the differences regarding the dimension and resolution for the different volumes in the combined dataset are illustrated. It is necessary to uniformize the data input to the network. This requires uniformization of the image spacing vector in all directions[3]. I chose to resample on a $1mm \times 1mm \times 1mm$ isotropic grid. If necessary, images are rotated to uniformize the orientation vectors. Every image is now a 3-dimensional array[4] with the same sequence of dimensions and the same spacing along these dimensions. The chosen uniform dimension sequence is:

0: Craniocaudal axis; perpendicular to the transverse plane.

1: Anteroposterior axis; perpendicular to the coronal (frontal) plane.

2: Left-right axis; perpendicular to the sagittal plane.

The inputs for the 2D models are obtained by *slicing* the volume along with one of the dimensions. This means that from each scan volume, three different sets of two-dimensional slices can be generated, depending on the slicing axis[5].

3. A typical CNN is not scale-invariant.

4. Both CT and MRI images have only one channel.

5. The slicing axis is the axis perpendicular to which ones slices the volume

When resampling the image itself, linear interpolation is used. To resample the classification masks on a new grid, the *nearest neighbour* method is used[6] since factorial data must not be interpolated.

6. SimpleITK [33] provides useful tools to perform these operations.

1.1.2 Contrast enhancement

To two-dimensional images obtained from slicing, the volumes are preprocessed with the Contrast Limited Adaptive Histogram Equalization (CLAHE) algorithm. The difference with ordinary histogram equalization is that the CLAHE method calculates different histograms for different sections of the image. This technique allows improving the local contrast in each region of an image. When an image contains regions that are significantly lighter or darker than the rest of the image (as is the case for both CT and MRI images), the AHE methods perform better than histogram equalization based on the complete image.

Figure 1.1: Image: 22^{th} sagittal slice of the 8^{th} image of the USiegen dataset. From left to right, the original image, the image after ordinary histrogram contrast enhancement and the result after CLAHE. The images show that CLAHE avoids increasing the noise in the larger dark areas while resulting in a more balanced image than the ordinary histogram method.

Contrast Limited Adaptive Histogram Equalization (CLAHE) is an improvement on adaptive histogram equalization. Limiting the contrast amplification avoids that noise is amplified in (near) constant regions of the image. The parameters of this operation were adjusted manually. Based on trial and error iterations, where the quality of the contrast improvement was estimated with the human eye based on visualizations from all datasets, the following parameters were chosen:

Number of grey bins: 256 (maximum possible when working with 8-bit encoded images).

Kernel size: 50. This parameter defines the extent of the *local* region used by the algorithm.

clip limit: 9.10^{-3}. This parameter suppresses noise in the darkest regions (which contain the least information).

1.2 Train, validation and test split considerations

To allow evaluation of the models produced, a test set is split from the development set. The objective is to evaluate how well a model generalizes to new data[7]. To provide an honest estimate of the out-of-sample performance, the test dataset should represent the investigated population and (hidden) correlations with elements of the development sample should be avoided.

This is done[8] taking into account the following:

7. New data is to be interpreted here as a new sample from *the same population* as the development sample. Unfortunately, I cannot claim that the different datasets I collected indeed form a perfect representation of the population of medical images of the lumbar spine.

8. To accomplish this, I used the function `GroupedStratifiedSplit` at `https://github.com/scikit-learn/scikit-learn/pull/18649/`. This class is not yet part of the official

Stratified data: The combination of data from different sources is stratified. Every source is considered a subpopulation. The data split is made such that the proportion of scans originating from each data source in every split is proportional to their occurrence in the total population.

Grouped data: The scans of the same patient can be assumed to be correlated to each other. These scans should not be spread over different splits. The data is split at patient level.

For each datasource, the intended split is $\frac{4}{6}$ for train set, $\frac{1}{6}$ for cross validation set and $\frac{1}{6}$ for test set. This distribution is not perfect but acceptable, as is indicated by the values in table 1.1.

split source	test	train	xval	total
MyoSegmenTUM	9	33	9	51
PLoS	4	14	4	22
USiegen	2	14	1	17
xVertSeg	2	10	3	15
total	17	71	17	105

Table 1.1: Number of volumes by datasource and by split. The scan volumes are split such that scans of the same patient are in the same split set.

split source	dimension	test	train	xval	total
MyoSegmenTUM	Transverse	1989	7293	1989	11271
	Frontal	1989	7293	1989	11271
	Sagittal	720	2650	725	4095
PLoS	Transverse	1528	5348	1528	8404
USiegen	Transverse	733	5063	501	6297
	Frontal	681	4276	1080	6037
	Sagittal	145	1089	180	1414
xVertSeg	Transverse	759	2652	785	4196
	Frontal	812	4163	1290	6265
	Sagittal	812	4163	1290	6265
total	Transvers	5009	20356	4803	30168
	Frontal	3482	15732	4359	23573
	Sagittal	1677	7902	2195	11774

Table 1.2: Number of slices by datasource and by split. These values might seem high, yet, the reader should not forget these image slices are highly correlated. There is very little additional independent information comparing one slice with a slice taken just 1mm further. On top of this, there are many slices which only contain the background class and do not provide much information for the model to train on the lumbar vertebrae classes.

The result of this operation is detailed in appendix A on page 78.

1.3 Cropping

The networks used require input images of size $352p \times 352p$, corresponding to $352mm \times 352mm$. The image slices have different dimensions. The incoming images are cropped or padded, depending on whether the image dimension is smaller or larger than the crop window.

When the image is cropped, one of 5 crops is selected, see figure 1.2:

crop 0: Crop from the top left corner of the image.

crop 1: Crop from the top right corner of the image.

crop 2: Crop from the bottom left corner of the image.

crop 3: Crop from the bottom right corner of the image.

crop 4: Center crop of the image.

Not all slices are sufficiently large to produce 5 different crops. This is illustrated in figure 1.3. In this case, one of the slice dimensions must be padded (symmetrically) to obtain an image of the desired dimensions.

When a slice is fetched from the train set, a random crop is selected from these 5. This adds an extra level of variance to the training data. This avoids, for example, that the network learns that the spine is always at the same location in the images.

For the slices sourced from the cross-validation and train set, however, reproducibility is essential. For each slice, the crop number is fixed. This way, the cross-validation metric, evaluated to avoid overfitting is always calculated on the same images.

When reconstructing the volumes, in the second step of the model concept (see chapter 2.1), all different crops are evaluated in the model, and the responses for these crops are joined by averaging the model responses in the overlap regions.

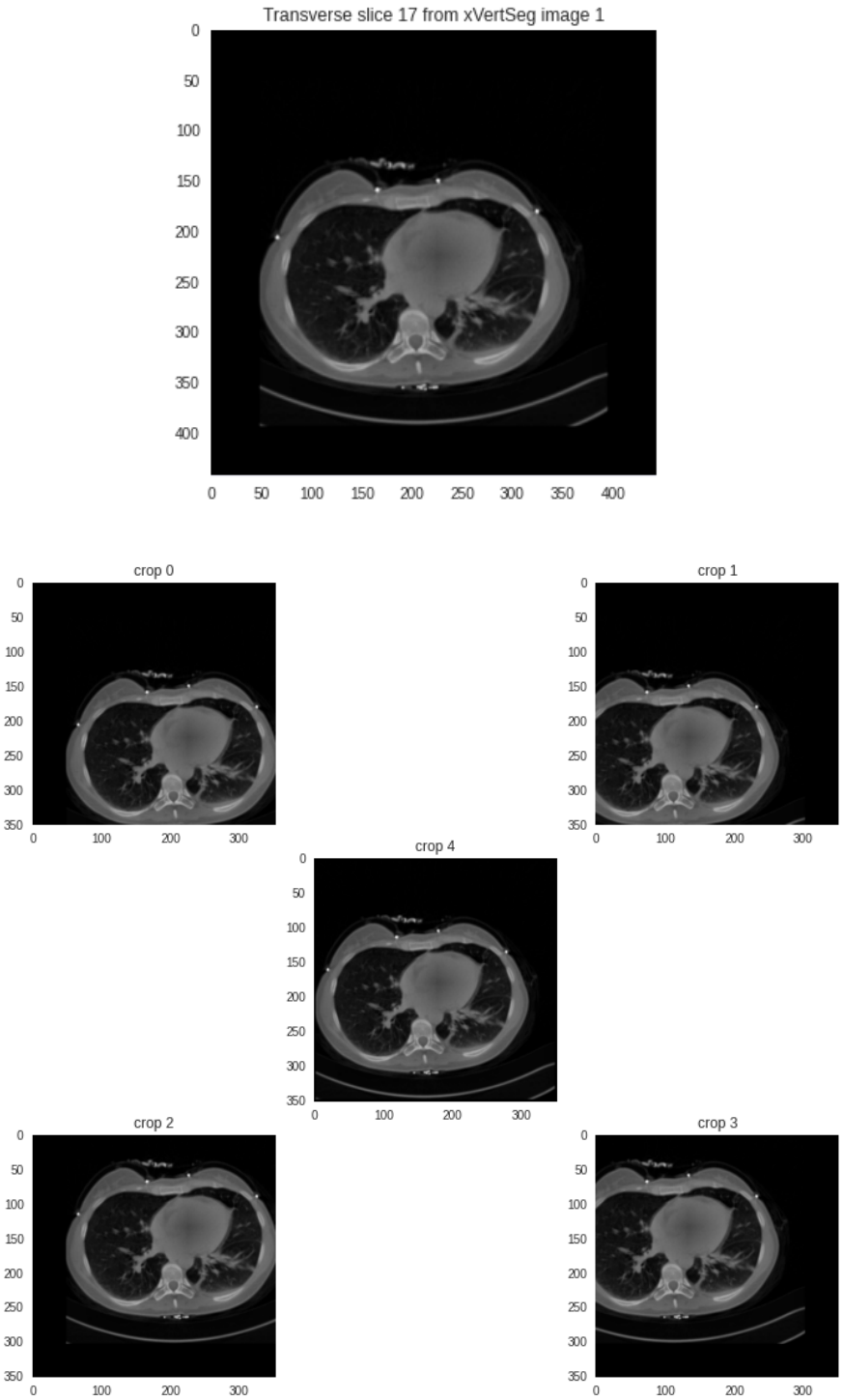


Figure 1.2: Illustration of image cropping where the image is larger than the crop window dimensions. The original size of the image is 443×443, and the crop window used has dimensions 352×352. Crop 0 to 4 are 5 different images but have a considerable overlap. This means that in chapter 4 on page 62, where the volumes are reconstructed from all relevant crops, for this type of slices four crops (0, 1, 2 & 3) have to be calculated.

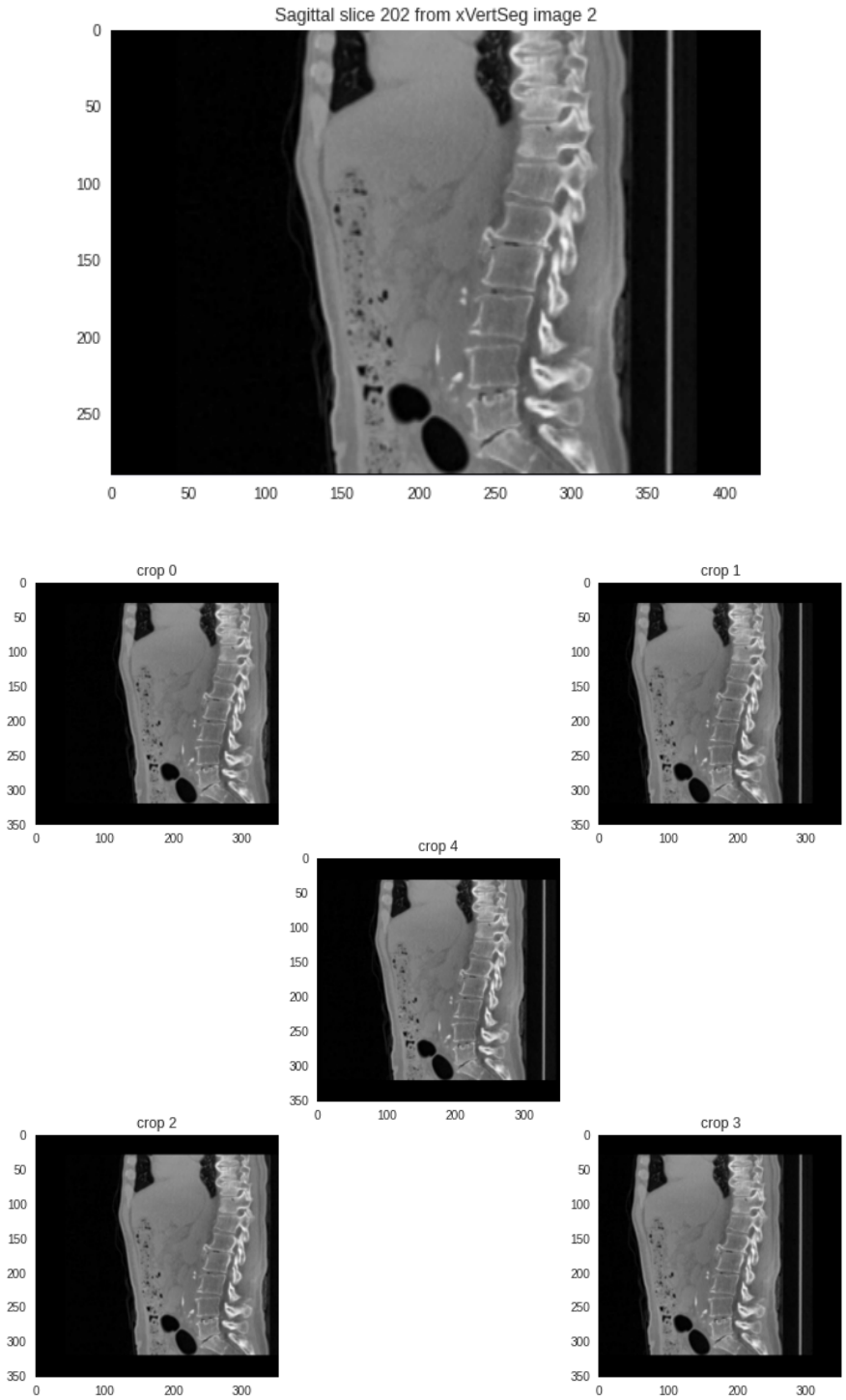


Figure 1.3: Illustration of image cropping where the image is larger than the crop window dimensions. The original size of the image is 291×424, and the crop window used has dimensions 352×352. Crop 0 to 4 are not all different images. Since the image height is lower than the crop window height, this dimension is padded, not cropped. This results in crop 0 being equal to crop 2 and crop 1 being equal to crop 3. This means that in chapter 4 on page 62, where the volumes are reconstructed from all relevant crops, for this type of slices only two crops (0 and 1) have to be calculated.

— 2 —

Modelling methods & evaluation protocols

This chapter introduces the high-level model concept of this work. This consists of producing a set of pseudo masks from three weakly supervised models trained on different sets of slices obtained from the same volume.

Next, the construction, training and evaluation of these weakly supervised models are covered in more detail. Attention is given to the model loss function, including two loss components that add to the consistency loss framework[22]. Finally, the model evaluation metrics are described.

These metrics will be used to compare the result of the Weakly supervised model with the result of a model trained on fully supervised labels. Both models will be trained on the same test scans and the model performances will be evaluated on the same test scans.

2.1 Modelling concept

2.1.1 Model type

The central model concept is the use of a 2D Convolutional Neural Network (CNN) to provide 3D segmentation masks, trained on point annotated data. Out-of-the-box pre-trained 2D networks are available, with weights pre-trained on public datasets[1]. An alternative approach would be to use 3D networks, which is a logical choice for volumetric data, with the apparent advantage that the 3D convolution layers naturally take into account the volumetric structure of the data. One could argue that using 2D networks deprives the network of crucial *context* information in the direction perpendicular to the slice it is analysing (this work tries to meet this issue in section 2.3.2 on page 33).

1. For example, weights trained on the ImageNet dataset. This dataset consists of over 14.10^6 images of more than 2.10^4 categories.

However, there are advantages to choosing 2D networks over 3D networks:

1. Since the number of parameters (weights) to train scales exponentially with the kernel dimension, the number of parameters to train is considerably higher for a 3D CNN compared to a corresponding 2D CNN.

2. The academic community has investigated 2D Convolutional Neural Networks for several years. The networks can be initialised with weights pre-trained on vast and diverse datasets (see remark 1 on ImageNet). These datasets indeed do not contain the specific classes nor the specific datatype investigated in this project. However, the different convolution layers have proven to extract useful general features that allow the network to distinguish an extensive range of categories. These features have proven to be sufficiently general to apply to new machine vision applications. This practice is called transfer learning.

3. Although consecutive slices are strongly correlated, one has more data[2] to train the networks. Certainly when considered in proportion to the number of weights to be trained.

2. There are more slices than volume segments in 1 image volume.

2.1.2 Model training approach

An approach used often in weakly supervised learning is the generation of *ersatz* or *pseudo* mask labels[19, 18, 1, 26], see page 18. This project intends to use this approach too.

A scan volume can be sliced along 3 different axes: the transverse axis, the coronal axis and the sagittal axis (see figure 1.2). This means that 3 different models can be trained with the available point annotated data. Each of these 3 networks will have different context elements to segment the sliced images on. The resulting segmentation volumes[3] of these three networks can then be combined to obtain a final segmentation volume. This final segmentation mask, which should be as least as good as the best of the 3 individual model results, can then be used as pseudo masks to train a final *fully supervised* network.

3. The segmentation results of a stack of 2D slices can be combined to form a 3D segmentation volume.

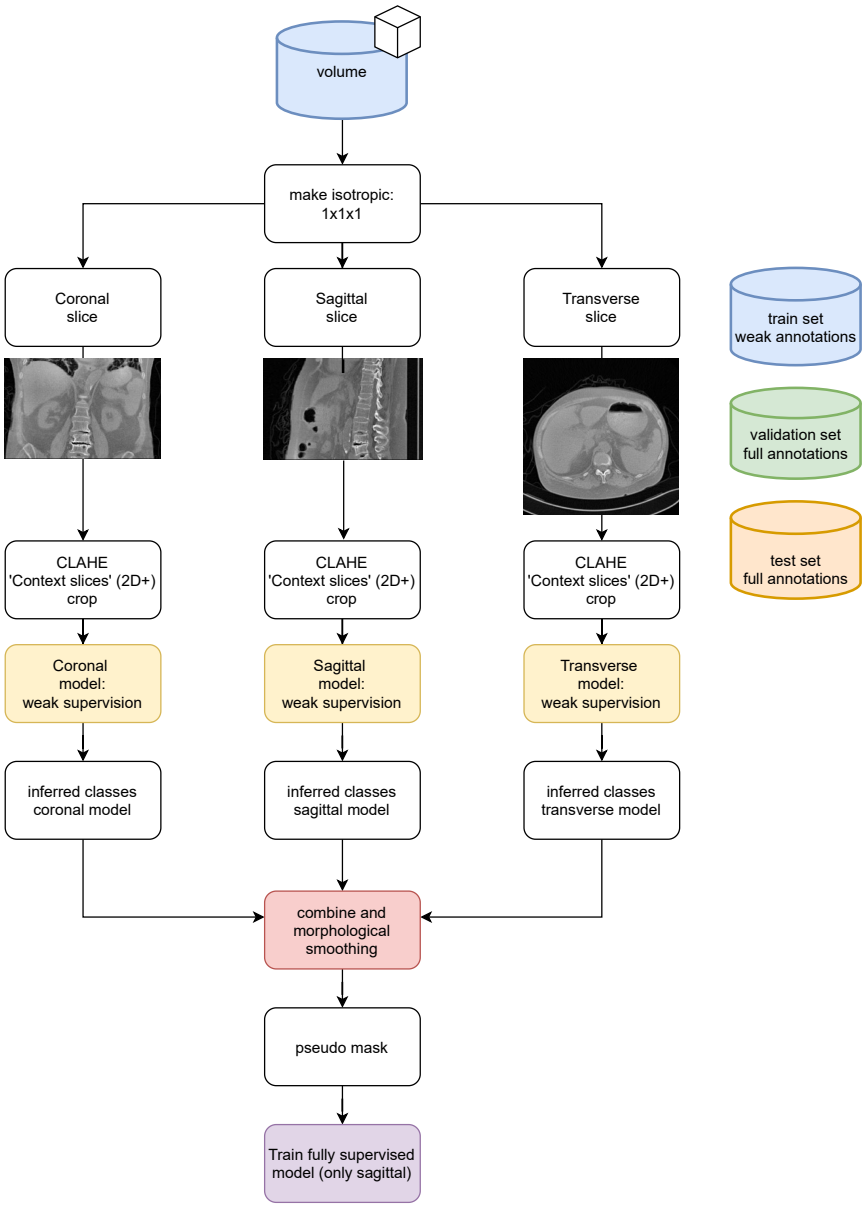


Figure 2.1: Illustration of the model training approach. This is based on the conbination of 3 different models based on different volume slices.

When the segmentation masks of a new, unknown volume need to be inferred, only this final model needs to be evaluated. Only one set of slices along a single dimension needs to be generated and preprocessed to perform this evaluation.

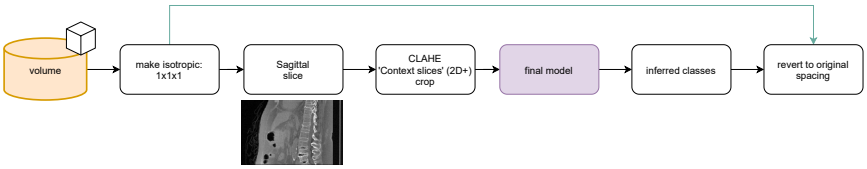


Figure 2.2: Inference step. Only one model needs to be evaluated in this step, the model that is trianed in the final step of the training procedure illustrated in figure 2.1.

2.2 Volume combination procedure

To construct one pseudo mask volume, first, three single dimension model are evaluated to form three stacks of two dimensional estimated segmentation masks, which are then combined to form three segmentation volumes. The resulting set of three segmentation volumes is then combined to form a new segmentation volume. It is this last segmentation volume, made out of the combination, that is sliced again to obtain the pseudo masks for the final model. The procedure described here was only devised after observation of the single dimensional model results.

2.2.1 Recombination of the crops to slices

All models are designed for 352×352 crops of the 2D slices[4]. To construct a segmentation volume from a single dimensional model, all relevant[5] crops of each volume slice are evaluated. First the segmentation results on these crops have to be combined again to obtain a segmentation mask of the whole slice. Due to the cropping procedure, different crops of the same slice always partly overlap. The crops are combined by averaging the logits z_i for the overlapping positions. The inferred class is then obtained from the resulting average z_i values for that position.

4. More information on the cropping of the slices can be found in chapter 1.3 on page 24.

5. Some slices have dimensions that do not allow to extract 5 different 352×352 crops. See for example figure 1.3 on page 27.

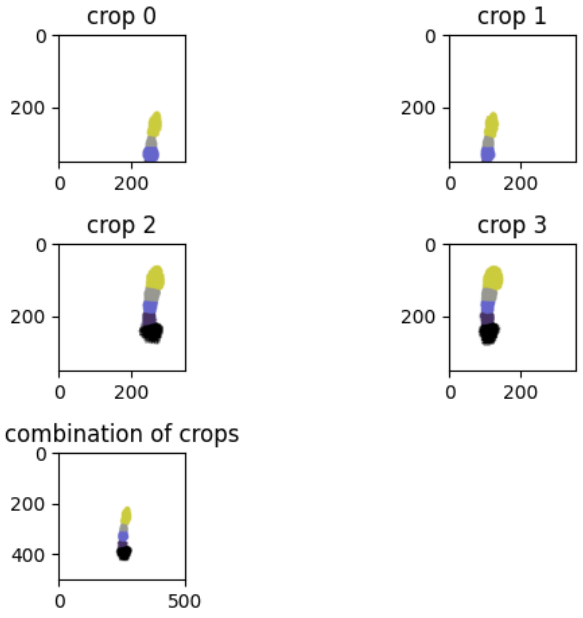


Figure 2.3: Sagittal slice 50 of the USiegen 04 volume has to be evaluated in 4 crops due to its dimensions. The recombination of these crops to form the mask for the slice is illustrated in this figure.

2.2.2 Rule based result combination

Once a stack of class segmentation masks for all slices of a volume are obtained, these can be combined to form a segmentation volumes. Combining the results of three different single dimension models is performed in two steps:

1. The resulting classification volumes are first combined with a rule-based method.
2. After this rule-based combination, the resulting segmentation estimation is smoothened with a morphological filter.

Three single dimension models are trained:

Transverse slices offer little context to indicate which of the lumbar vertebrae they contain. It does not seem easy even for a human expert to indicate which vertebra is visible on the slice. The model trained on these slices is intended only for semantic segmentation. Each pixel is inferred only if it represents a vertebra, without distinction between the different lumbar vertebrae.

Sagittal & Coronal slices do offer the necessary context to distinguish between L_1 to L_5. The models trained on these slices do indicate the specific lumbar vertebra index.

The volume combination rules are based on two observations:[6].

1. The precision (see equation 2.12 on page 41) with which the background class is predicted in all models is very high. This means $\mathcal{P}\left(label = background \mid prediction = background\right)$ is high for all models. If there is one model that indicates a position is background, this position could be estimated to be background with high probability.
2. All models tend to over-estimate the extend of the vertebrae. This causes a low recall score. The direction in which the predictions tend to *bleed out* is different for the different models.

The observations mentioned above can be combined in algorithm 1.

6. The resulting estimations from a model for an input volume is again a three-dimensional volume ($\in \quad \mathbb{N}^3$) with the same dimensions.

2.2.3 Morphological smoothing

After the rule-based combination of estimations from different single dimension models, the result is smoothened with standard morphological filters. These filters are combinations of the morphological *erosion* and *dilation* operators[7].

- Noise in the single dimension segmentation masks is suppressed with an opening operation on the individual segmentation volumes of the single dimension models.
- Noise in the combined volumes is suppressed by first opening and then closing the volumes.
- The estimated volumes are observed to overestimate the extent of the vertebrae. For this reason, an erosion step is performed to decrease the overall extent of the class masks.

7. This document is not intended to provide an elaborate explanation on morphological operations. For the readers conventience, the following symbolic notations are repeated:

Erosion of set $\mathbf{A}$ by structural element $\mathbf{B}$: $\mathbf{A} \ominus \mathbf{B}$

Dilation of set $\mathbf{A}$ by structural element $\mathbf{B}$: $\mathbf{A} \oplus \mathbf{B}$

Opening of set $\mathbf{A}$ by structural element $\mathbf{B}$: $\mathbf{A} \circ \mathbf{B} = (\mathbf{A} \ominus \mathbf{B}) \oplus \mathbf{B}$

Closing of set $\mathbf{A}$ by structural element $\mathbf{B}$: $\mathbf{A} \bullet \mathbf{B} = (\mathbf{A} \oplus \mathbf{B}) \ominus \mathbf{B}$

The procedure mentioned above has two hyperparameters: the number of iterations for the denoising filters and the number of iterations for the erosion filter. Both hyperparameters are estimated by calculating the same evaluation metric, the weighted dice score on the validation set, as for the single-dimensional model evaluation.

2.2.4 Volume combination algorithm

The combination of the rules and morphological smoothing operations discussed above result in the following algorithm to combine the segmentation volumes from the single dimension models[8]:

8. The algorithm shown here is based on the two observations mentioned.

Algorithm 1: Rule based combination of model results from three single dimension models

Data: Results $y_{\cdot}$ of three models indicating an estimated class for all positions $\vec{p}$ in the volume. ;
Transverse model $y_t \in \mathbb{N}^3 : \forall y_t(\vec{p}) \in \{0, 1\}$;
Sagittal model $y_s \in \mathbb{N}^3 : \forall y_t(\vec{p}) \in \{0, 1, 2, 3, 4, 5\}$;
Coronal model $y_c \in \mathbb{N}^3 : \forall y_t(\vec{p}) \in \{0, 1, 2, 3, 4, 5\}$;
Binary $\mathbf{B}$ structure with rank 3 and connectivity 3 ;
Result: Combination of the three model results y_f.
`// Closing operation on all` $y_{\cdot}$
$y_t \leftarrow y_t \bullet \mathbf{B}$;
$y_s \leftarrow y_s \bullet \mathbf{B}$;
$y_c \leftarrow y_c \bullet \mathbf{B}$;
for *all* $\vec{p}$ **do**
  $i \in [1, 2, 3, 4, 5]$;
  if $y_t[\vec{p}] = 1 \wedge y_s[\vec{p}] = i \wedge y_c[\vec{p}] = i$ **then** $y_f[\vec{p}] \leftarrow i$;
  else $y_f[\vec{p}] \leftarrow 0$;
end
$y_f \leftarrow ((y_f \circ \mathbf{B}) \bullet \mathbf{B}) \ominus \mathbf{B}$

2.3 Model hyperparameters

Building the different models for the procedure illustrated in figure 2.1 requires making several choices. In this section, components and hyperparameters between which were chosen are introduced.

2.3.1 Network types

Three different Convolutional Neural Network architectures were tested for this project. All three of these networks are based on the encoder-decoder principle where a contracting part of the network extracts Features and an expanding layer sequence produces a segmentation mask.

VGG16 network: This is a classic classification network, developed by K. Simonyan & A. Zimmerman. The used network is illustrated in figure 2.4. Inspired by [22], the network is adapted for segmentation (This segmentation architecture is called VGG16 FCN8) by omitting the flattining layer for image level classification and replacing it with an upsampling path. In this upsampling path, the (expanded) pooling layers after conv3 and conv4 are added to the results with element-wise summation.

U-Net network: The U-Net is a network specifically designed for segmentation tasks. As was already described in chapter 3.1 on page 15, this network architecture was originally published in [27].

RESNET50 network: The Residual Network (resnet) (Kaiming He et al.) is a classification architecture that can be made very deep (more than 150 layers). This depth is possible thanks to the skip connections. The network tested in this work is an implementation of the RESNET50 architecture, adapted with a FCN8 upsampling segmentation part. The RESNET50 architecture can be seen as a stack of *residual units*. In figure 2.4, four such a residual block is illustrated. Every few residual units, the number of feature maps is doubled while the height and width are halved. When this happens, the input cannot be added directly to the output of the residual unit. To solve this problem, the inputs are passed through a 1×1 convolution layer with stride 2 and the right number of output feature maps (the blue layer in figure 2.4).

2.3.2 2D+ approach and *context slices*

As discussed in section 2.3.1, all used networks have a 3 channel[9], 352×352 pixel input. The input images for the network only have 1 channel component. The other two channels were used to pass *context* information to the network in the form of the neighbouring slices of the slice under investigation. This combines an essentially 2D approach with a third dimensional element in the form of the extra information of the parallel slices. Thus, this is called the $2D+$ approach.

The hyperparameter to be set k is the index offset for the context slices. When evaluating slice with index n the slices with indices $[n-k; n+k]$ are passed as context slices [10].

Several values for k are tested. When $k = 0$, no extra slices are taken. The same slice is just entered in all 3 network channels. When $k = 1$, the indices directly next to the investigated slices are taken. Due to the isotropic resampling (see section 1.1.1), these slices are physically at 1 mm distance from the centre slice. When $k = 5$, the slices at $5mm$ distance from the centre slice are used.

This stack of 3 slices is then passed through the network, as illustrated in figure 2.4.

9. The networks were designed initially for *RGB*-images with a red, a green and a blue channel.

10. For the edge cases ($n < k$ or $n + k >$ image dimension) when these slices do not exist, slice n is passed as context slice.

2.3.3 Annotation points

An important hyperparameter is the number of annotation points per instance slice. This consists of two parameters: the background annotation points and the class annotation points. In this work, these are generated from the full masks provided in 4 of the 5 datasets. In every slice, annotation points are samples from the available masks in this slice.

This project aims to investigate how the performance of a segmentation model trained on fully supervised data can be approximated with a model trained on weakly supervised (point annotated) data. In the first place, this will be investigated by comparing models trained on the fully annotated data to models trained only with point labels generated based on the full masks. The generation of these point labels is automated. The desired number of

point labels is generated from the full segmentation ground by random selection of points from these masks. I did not investigate how closely point annotations generated in this way resemble point annotations generated by a human expert.

In the first part of the research, the aim is to evaluate the difference between a model trained on a fully labelled set of images and a model trained on the same set of images for which only weak labels are provided. For this purpose, annotation points are generated for each slice. The individual models are thus trained with a fixed number of points per class instance in each slice. This approximates the situation where an expert annotates not one but three piles of slices, one for each dimension. This is not the final objective of this research. Based on the results obtained from this first step, three models are now trained based on a single set of annotation points, obtained from the annotation of slices along one dimension of the volume. Apart from the slices along the chosen dimension, the number of annotation points in the slices along the other two dimensions are not fixed.

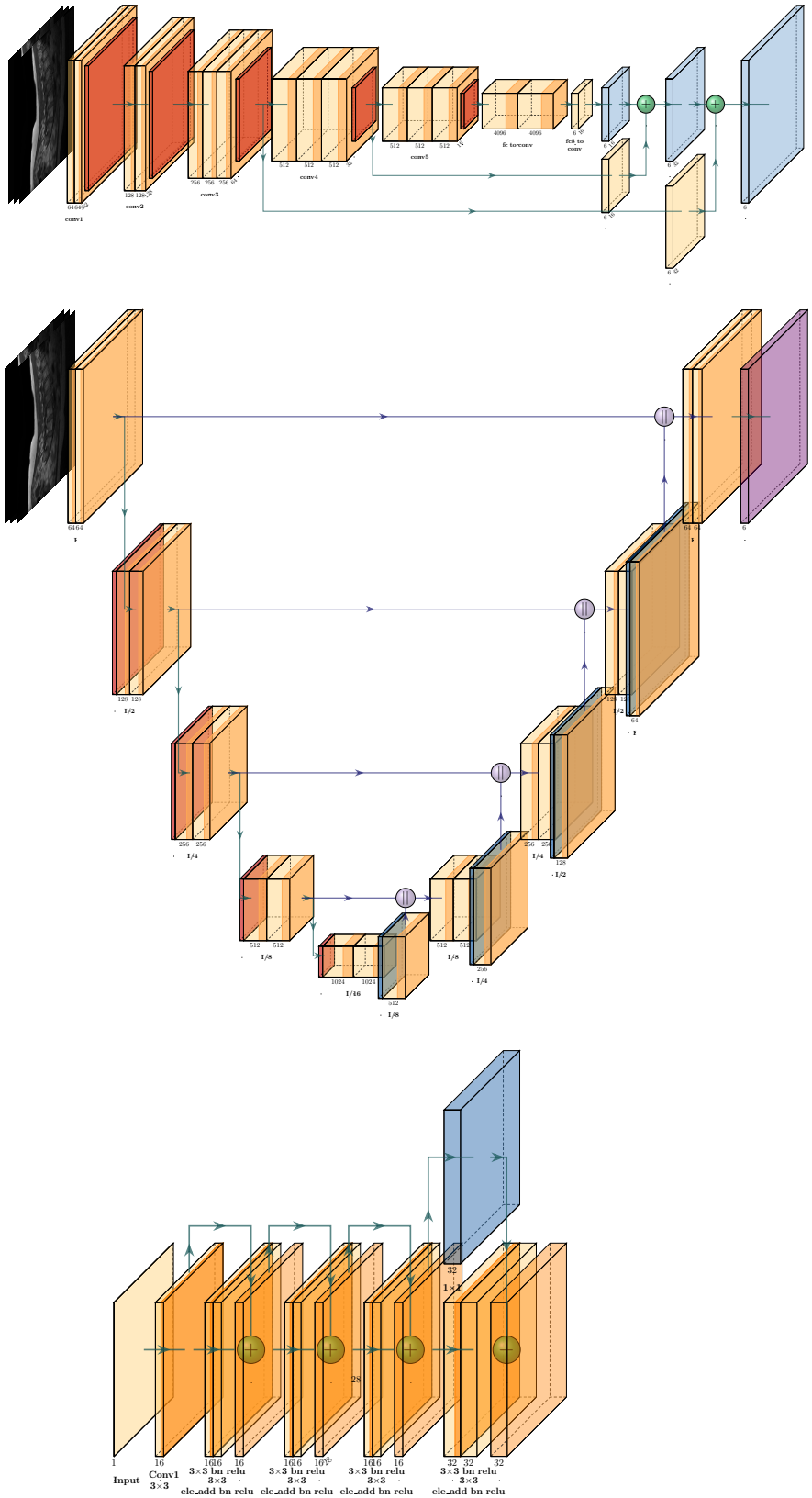


Figure 2.4: Illustrations of the tested network architectures. The first network is the VGG16-FCN8 network.

Then the U-Net architecture is illustrated.

Finally, one residual unit of the RESNET50 architecture is illustrated. Due to the size of the RESNET50 network, only one *residual block* is illustrated.

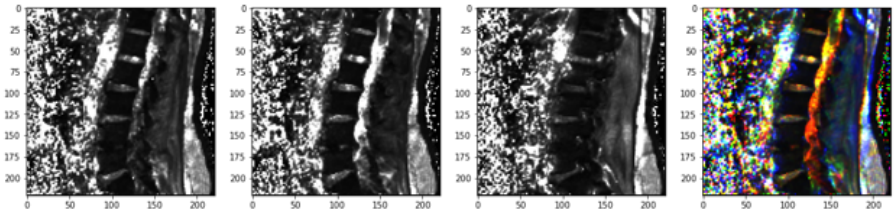


Figure 2.5: Comparison and image slice (MyoSegmenTUM dataset) with its context slices at 3mm distance. The 4^th image shows the combination of the three first as R, G and B channels of a colour image to illustrate the local difference. It is clear that these images are highly correlated but do hold some additional information.

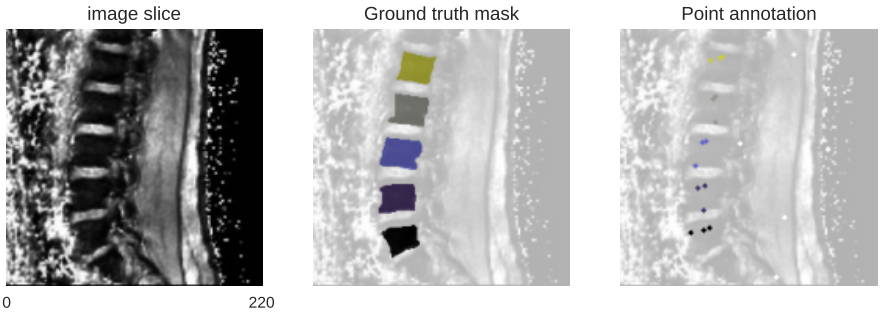


Figure 2.6: This image illustrates how the Ground truth mask for MyoSegmenTUM image 020 (sagittal slice 21) is converted from a full annotation mask to a point annotation mask.
Colour legend:
● L1 ● L2 ● L3 ● L4 ● L5

2.3.4 Model training strategy

To optimize the model weights, the optimization scheme in algorithm 2 is executed. The goal is to obtain model weights θ_b which show the best model performance, measured with metric $\mathcal{M}$ on the validation set $\mathcal{S}_{val}$. This is done by evaluating the model with a pre-defined set of learning late reduction factors $\vec{f}_{LR}$ until the chosen metric $\mathcal{M}$ does not decrease anymore with decreasing training loss[11].

Algorithm 2: Model optimization strategy. The metric $\mathcal{M}$ used in this algorithm is the inverse class weighted dice score, etailed in equation 2.18 on page 42.

Data: Train set $\mathcal{S}_{train}$ with weak annotations ;
Validation set $\mathcal{S}_{val}$ with full annotations ;
Test set $\mathcal{S}_{test}$ with full annotations ;
Result: Trained model weights θ_b ;
Train, cross-validation and test metrics ;
Algorithm:
$\mathcal{M}_b \leftarrow -\infty$ (best metric);
$\mathcal{L}_b \leftarrow \infty$ (best loss);
$LR \leftarrow 10^{-4}$;
$\vec{f}_{LR} \leftarrow [1, 10, 10, 50]$;
$T_{\mathcal{M}} \leftarrow 0$;
$T_{\mathcal{M},max} \leftarrow 10$;
for $fact_{LR} \in \vec{f}_{LR}$ **do**
  $LR \leftarrow LR/fact_{LR}$;
  $T_{\mathcal{L}} \leftarrow 0$;
  while $T_{\mathcal{L}} < T_{\mathcal{L},max}$ **do**
    $\mathcal{L}_n \leftarrow$ train epoch($\mathcal{S}_{train}$) ;
    $\mathcal{M}_n \leftarrow$ evaluate($\mathcal{S}_{val}$) ;
    if $\mathcal{L}_n < \mathcal{L}_b$ **then**
      $T_{\mathcal{L}} \leftarrow 0$;
      $\mathcal{L}_b \leftarrow \mathcal{L}_n$;
      if $\mathcal{M}_n > \mathcal{M}_b$ **then**
        $T_{\mathcal{M}} \leftarrow 0$;
        $\mathcal{M}_b \leftarrow \mathcal{M}_n$;
        $\theta_b \leftarrow$ current model weights ;
      else
        $T_{\mathcal{M}} \leftarrow T_{\mathcal{M}} + 1$;
      end
    else
      $T_{\mathcal{L}} \leftarrow T_{\mathcal{L}} + 1$;
    end
  end
end
set model weights to θ_b ;
evaluate all metrics on $\mathcal{S}_{train}$, $\mathcal{S}_{train}$ & $\mathcal{S}_{test}$;

11. This is tracked with the counter $T_{\mathcal{M}}$. Every time a training loss $\mathcal{L}_n$ is obtained lower than the best training loss $\mathcal{L}_b$ obtained up to that point, the performance metric is compared to the best performance metric obtained up to that point. If not improvement of the model performance metric can be observed $T_{\mathcal{M},max}$ times in a row, the model training is halted, and the model weights are restored to θ_b, the best performing set of weights. Evaluating more epochs would only result in overtraining the model to $\mathcal{S}_{train}$.

There is also a second counter: $T_{\mathcal{L}}$. If the optimizer is not able to decrease the model loss further with a given learning rate for a given number of epochs $T_{\mathcal{L},max}$, the learning rate will be decreased with a factor defined in $\vec{f}_{LR}$. Since $\vec{f}_{LR}$ is finite, it is indeed possible that the optimization will be stopped because the model is still not capable of decreasing the loss at the last learning rate reduction.

The used optimization algorithm is *Adam*, or Adaptive Moment estimating. This is a widely used and performant optimizer that has proven to work well in practice. This optimizer is started with a learning rate of 10^{-4}. The learning rate can be reduced several times during the optimization procedure.

In algorithm 2, the full optimization procedure is described.

2.4 Loss functions

Construction of a machine learning model requires evaluating the model, comparing the result of this evaluation with the desired output and taking steps to make the observed output resemble the desired output better. In other words, to train a model, the optimizer will minimize the loss function. To use the gradient descent algorithm, this loss function needs to be differentiable.

Several loss components will be discussed below, both unsupervised and supervised loss functions. A supervised loss function component considers the difference between the network prediction and a known (weak) label. An unsupervised loss function component considers characteristics of the network output itself. No labels are used in calculating an unsupervised loss component. The desirable characteristics that are enforced by the unsupervised loss functions can be part of the *priors*, the knowledge on the problem one has beforehand [12]. In order to make the notation in this section clearer, table 2.1 list the notations used in the loss function components.

12. To be able to work with weaker, less informative labels, one needs to enter more prior knowledge into the problem

Symbol	meaning
n	Number of problem classes
X_i	Slice i and corresponding context slices
$\vec{p}$	A pixel position in slice i
$\mathcal{K}$	Set of outcome classes: $\mathcal{K} = [0, 1, \ldots, n-1]$
$\mathcal{Y}_i$	set of labels for slice i, $\mathcal{Y}_i(\vec{p}) \in \mathcal{K}$
$\mathcal{I}_i$	set of pixel positions for which a label is available
$f_\theta(X_i) = z_i$	model output (logits) for X_i with weights θ. $z_i(\vec{p}) \in \mathbb{R}^n$
$\sigma(z_i(\vec{p}))$	softmax result for $z_i(\vec{p})$ ($\in \mathbb{R}^n$ and $\forall k \in \mathcal{K} 0 \leq \sigma_k \leq 1$)
w_k	A weigth that can be attributed to class $k \in \mathcal{K}$

Table 2.1: List of symbols used in the loss functions

In this project, $n = 6$ for the coronal and sagittal slice models and $n = 2$ for the transverse slice models. Label $\mathcal{Y}_i(\vec{p}) = 0$ indicates the background class and the lubar vertebrae L_m are indicated with $\mathcal{Y}_i(\vec{p}) = m$ ($m \in [1, 2, 3, 4, 5]$). In the fully supervised case, $\mathcal{I}_i$ spans the complete slice i. In the weakly supervised case, there are only a handful of positions $\vec{p}$ for which a label exists. The softmax function σ is related to the sigmoid function $\mathbf{S}$. It is defined as:

$$\sigma_k(z) = \frac{\exp(z_k)}{\sum_{m \in \mathcal{K}} \exp(z_m)} \tag{2.1}$$

$$\mathbf{S}(x) = [1 + exp(-x)]^{-1} \tag{2.2}$$

$$= \frac{exp(x)}{exp(x) + 1} \tag{2.3}$$

The final training loss is given by:

$$\mathcal{L} = \mathcal{L}_P + \mathcal{L}_E + \mathcal{L}_C + \mathcal{L}_S \tag{2.4}$$

The different loss components are detailed below.

2.4.1 Supervised loss functions

cross entropy loss & Point loss

To compare the network output with the labels, first, the (weighted) cross entropy loss is used[13].

$$\mathcal{L}_P(X_i) = -\sum_{\vec{p}\in\mathcal{I}_i} w_{\mathcal{Y}_i(\vec{p})}.\log\left[\sigma_{\mathcal{Y}_i(\vec{p})}\left(z_i(\vec{p})\right)\right] \tag{2.5}$$

This loss term is minimized when the network output $z_i(\vec{p})[k]$ with $k = \mathcal{Y}_i(\vec{p})$ the class label for $\vec{p}$ is maximal while the other logits for this position are minimal. In the fully supervised case, this loss function is called the cross entropy loss. When training a model with point supervision, this is only one of multiple loss components. In the case of point supervised data, $\mathcal{I}_i$ only consists of a handful of points. This loss component is in this case called the point loss component $\mathcal{L}_P$.

Prior extend loss

The second Supervised loss function used is the *prior extend* loss. This loss function describes the prior knowledge that the dimensions of a vertebra are limited. Based on [2], the maximal extent of a human lumbar vertebrae was set to be $r = 110mm$. This means that a point label $\mathcal{Y}_i(\vec{p}) = m : m \in [1, 2, 3, 4, 5]$ indicating that $\vec{p}$ is a point in vertebra L_m, also means that all points outside of a cicle with radius r cannot be points of L_m. First, $n-1$ (in this case 5) distance masks are created[14], representing the maximal[15] distance (Euclidian norm) of each location $\vec{q}$ from the label $\mathcal{Y}_i(\vec{p}) = k$ furthest away from $\vec{q}$. Then $\mathbf{d}$ is converted to a semi-mask:

$$\mathbf{d}_k(\vec{q}) = \max_{\vec{p}:\mathcal{Y}_i(\vec{p})=k} ||\vec{q}-\vec{p}|| \tag{2.6}$$

$$\mathbf{m}_k(\vec{q}) = \mathbf{I}\left((-\mathbf{d}(\vec{q})+r) > 0\right) \tag{2.7}$$

Now, $\mathbf{m}$ is 1 for positions closer than distance r from the points, and it is 0 for positions far from labels with value k. Where $\mathbf{m}_k = 0$, the model output should not indicate output class k. Where $\mathbf{m}_k = 1$, the output class is unknown[16].

The loss function is the binary cross-entropy between $\mathbf{m}_k$ and the sigmoid of the kth channel of the logits z_i with weight vector $\{1, 0\}$.

$$\mathcal{L}_E(X_i) = \sum_{k\in\mathcal{K}}\sum_{\vec{q}\in X_i}(1-\mathbf{m}_k(\vec{q}))\log(\mathbf{S}(z_i(\vec{q})_k)) \tag{2.8}$$

2.4.2 Unsupervised loss functions

Transformation consistency loss

The first Unsupervised loss function is the unsupervised consistency loss[22]. The idea behind this loss is that the model output should be consistent for geometric transformations of the input. A set of geometrical transformations $T = \{t_1, t_2, \ldots, t_n\}$ is defined. Then the consistency loss function $\mathcal{L}_C$ is defined as [17]

$$\mathcal{L}_C(X_i) = \sum_{p\in\mathcal{P}_i}\left|t_k\left[f_\theta(X_i)\right]_p - f_\theta\left(t_k[X_i]\right)_p\right| \tag{2.9}$$

.

13. The weight is optional. One can decide $\forall i \in \mathcal{K} : w_i = 1$

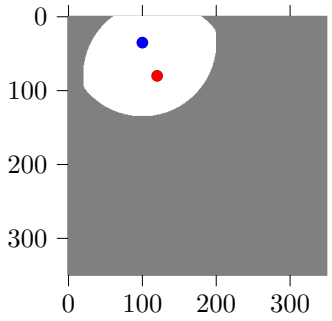


Figure 2.7: Illustration of the prior extend mask for annotation points at [120, 80] and [100, 34]. In the grey area, $\mathbf{m} = 0$.

14. 1 label is background, for which there is no prior extend.

15. Maximal distance, because there can be mulitple such labels.

16. Only channel k is concerned, since one only knows what class these points do not belong to, there is not more information about the other classes.

17. This function evaluates the difference between the transformation of the model output $t_k\left[f_\theta(X_i)\right]$ and the model output for the transformed input $f_\theta\left(t_k[X_i]\right)$.

In this project, two transformations are combined: rotations of $[90°, 180°, 270°]$ and the horizontal flip. Each time the loss is called, 1 transformation t is randomly selected from the list, and the transformation consistency loss is calculated using this transformation.

Separation loss

As mentioned when discussing the cross-entropy loss in §2.4.1, this loss encourages the model to output the correct label and to decrease the output values for the other labels. As discussed, the right label is often not know for most of the locations in a weakly supervised problem. Apart from a limited number of points, there is no such incentive for weakly supervised networks. For this reason, the separation loss $\mathcal{L}_S$ is introduced. This loss component consists of the negative sum of the absolute difference of the class segmentation masks[18]:

$$\mathcal{L}_S(X_i) = -\sum_{\vec{p}} \sum_{m \in \mathcal{K}} \sum_{n \in \mathcal{K}, n > m} \mathbf{S}(z_i[m]) - \mathbf{S}(z_i[n]) \tag{2.10}$$

18. $z_i[m]$ denotes the m^{th} model output channel.

This loss component thus provides an incentive to the network to make a decision, even in areas where few annotation points are available.

2.5 Metrics

Metrics are used to evaluate and compare Machine Learning models. This text focuses on the important metrics used for multi-class classification problems.

The example in table 2.2 will be used to introduce the different metrics. Table 2.2 illustrates a confusion matrix for a model for a problem with 3 classes ($C_j : j \in \{1; 2; 3\}$). A model predicts the class for all observations in the set. The confusion matrix allows comparing these predictions against the true class[19]. $a_{0,0}$, $a_{1,1}$, $a_{2,2}$ are the correctly labelled observations. The predicted class corresponds to the actual class. This is not necessarily the case for all observations. For example, $a_{2,1}$ observations with true class C_2 the model predicted to belong to class C_1. In general, $a_{i,j}$ is the number of observations for which the model predicted C_j and for which the label is C_i. Based on the confusion matrix, 3 metrics can be calculated: the model *precision*, the *recall* and the *F-score*. These are illustrated in figure 2.8.

		Predicted class		
		C_0	C_1	C_2
Actual class	C_0	$a_{0,0}$	$a_{0,1}$	$a_{0,2}$
	C_1	$a_{1,0}$	$a_{1,1}$	$a_{1,2}$
	C_2	$a_{2,0}$	$a_{2,1}$	$a_{2,2}$

Table 2.2: Illustration of confusion matrix with 3 classes.

19. The true class is the label for this observation.

2.5.1 Class imbalance

To construct an appropriate set of metrics to use for a problem, the class distribution of the dataset[20] needs to be taken into account. Imagine, for example, that one aims to segment pixels in a set of pictures where 95% of the pixels is of the *background* class and 5% of the pixels is of the class to be segmented. A model can now quickly obtain 95% accuracy by predicting every pixel as *background*. This is not a metric suitable for the problem.

For an evaluation metric to be useful, it has to avoid the kind this kind of pitfall. In this work, I investigate a problem where most of the image under investigation is *background*. There is about 1 pixel of each of the lumbar vertebra classes for 500 background pixels. The different performance metrics for the different classes are weighted with the inverse of their relative occurrence to evaluate the model's performance correctly.

20. One hopes that through a good data collection strategy, the dataset represents the population on which we ultimately want to infer.

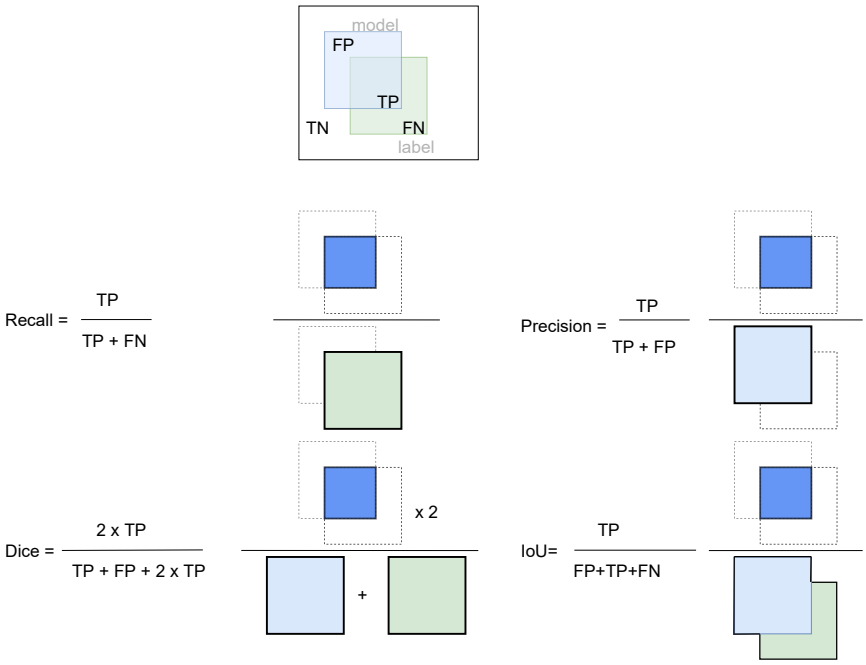


Figure 2.8: Illustration of different evaluation metrics. The observations for which the model correctly predicts a positive label are called the *True Positive* (TP) values. The observations for which a positive label is wrongfully predicted are the *False Positive* (FP) labels. When an observation is positive, but the model fails to predict this, this is a *False Negative* (FN) observation. On observation that is negative and is correctly predicted to be by the model is a *True Negative* (TN) observation.

2.5.2 Precision

The **precision**[21], for class i,[22] is the proportion of true labels C_i out of all observations predicted to be C_i. For example, the precision for class 1 in table 2.2 is:

$$\text{Precision}_1 = \frac{a_{1,1}}{a_{0,1} + a_{1,1} + a_{2,1}} \qquad (\text{Precision}_1 \text{ in table 2.2})$$

. In general, the Precision_i for class C_i in a problem with k classes is given by equation 2.12.

$$\begin{aligned} \text{Precision}_i &= \mathcal{P}\left(label = C_i \mid prediction = C_i\right) && (2.11) \\ &= \frac{a_{i,i}}{\sum_{j=0}^{k-1} a_{j,i}} && (2.12) \end{aligned}$$

One can thus calculate the precision metric for each class. To calculate a precision metric for the complete multi-label classifier , one can either aggregate the class precisions by taking the arithmetic mean (this is called the Macro-precision: $Precision_M$) or by taking the weighted mean (this is called the weighted-mean precision: $Precision_w$). To take the weighted mean, each precision term $Precision_i$ is weighted by the number of observations with label C_i. Thus, more importance is given to classes with higher occurrence.

$$\begin{aligned} \text{Precision}_M &= \frac{\sum_{i=0}^{k-1} \text{Precision}_i}{k} && (2.13) \\ \text{Precision}_w &= \frac{\sum_{i=0}^{k-1} \left[\text{Precision}_i \sum_{j=0}^{k-1} a_{i,j}\right]}{\sum_{i=0}^{k-1} \sum_{j=0}^{k-1} a_{i,j}} && (2.14) \end{aligned}$$

Recall

The **recall**[23] is the numer of correctly predicted C_i observations out of the total number of C_i observations. For example, the recall for class 1 in table

21. the precision is also called the *positive predictive value*.

22. Note that for binary classifiers (reject a null-hypothesis H_0 or do not reject H_0) the metrics for the positive class are understood to be the classifier metrics. One will traditionally report the classifier precision as $\frac{TP}{TP+FP}$ without calculating the precision for the negative class where H_0 is not rejected. The binary confusion matrix clarifies the meaning of TP (True Positive) and FP (False Positive).

		Pred. H_0	**Pred.** $\neg H_0$
Actual	H_0	TN	FP
	$\neg H_0$	FN	TP

.

23. The recall is also called the *sensitivity*.

2.2 is:

$$\text{Recall}_1 = \frac{a_{1,1}}{a_{1,0} + a_{1,1} + a_{1,2}} \qquad (\text{Recall}_1 \text{ in table 2.2})$$

. The general expression for recall_i is equation 2.16.

$$\begin{aligned} \text{recall}_i &= \mathcal{P}\left(prediction = C_i \mid label = C_i\right) && (2.15) \\ &= \frac{a_{i,i}}{\sum_{j=0}^{k-1} a_{i,j}} && (2.16) \end{aligned}$$

The weighted-mean recall and macro-recall are defined in the same way as the multi-label precision metrics, see equation 2.13 and equation 2.14.

2.5.3 Dice score

The objective is, of course, to build a model with both high precision and a high recall. There is often a trade-off to be made. Increasing the recall tends to reduce the precision [24]. One needs to find a balance between both.

It is useful to combine both metrics in a single new metric: the **F1-score**[25]. This is accomplished by taking the harmonic mean [26] of precision and recall.

$$\text{Dice}_i = 2.\frac{precision_i \times recall_i}{precision_i + recall_i} \qquad (2.17)$$

Calculation this for Dice_1 in the example shows the dice score can also be described as double the intersection between the points predicted as C_1 and the points with true label C_1 devided by the sum of the number of points predicted as C_1 and the number of points with true label C_1.

$$\text{Dice}_1 = \frac{2.a_{1,1}}{a_{0,1} + 2.a_{1,1} + a_{2,1} + a_{1,0} + a_{1,2}} \qquad (\text{Dice}_1 \text{ in table 2.2})$$

Now, there are several options for aggregation in the multi-class case: There are two macro F1-scores in use:

macro F1-score : Calculate the F1-score for each class and take the arithmatic mean of the k class F1-scores. As discussed in §2.5.1, for inbalanced problems, a weighted mean can help to more acurately describe the model quality.

macro F1*-score : Calculate the class precision and recall scores. From these class precision and recall scores, calculate the macro precision and macro recall score. The macro F1*-score is given by $F1^*_M = 2.\frac{precision_M \times recall_M}{precision_M + recall_M}$

weighted F1-score : Calculate the F1 score for all classes and calculated the weighted average as is done in equation 2.14 to calculate the weighted precision.

The *inverse* class weighted F1-score or Dice score is used to evaluate the experiment results in algorithm 2 and in part III starting at page 44. The expression defining Dice_{wi} is iterated below[27]:

$$\text{Dice}_{wi} = \frac{\sum_{i=0}^{k-1} \left[\text{Dice}_i \left(\sum_{j=0}^{k-1} a_{i,j}\right)^{-1}\right]}{\sum_{i=0}^{k-1} \left(\sum_{j=0}^{k-1} a_{i,j}\right)^{-1}} \qquad (2.18)$$

The objective of using the inverse of the label occurrence is to increase the importance of the underrepresented classes in unbalanced data.

24. To increase recall_i, you need to encourage the model to predict C_i. Unfortunately, this increases the probability that an observation is wrongly classified as C_i, thus decreasing precison_i.

25. The F1-score is also called the Dice-score

26. The harmonic mean assures that F1 will always be between the values of precision and recall, but it will be closer to the lowest value. If, for example, $recall = 0$ and $precision = 1$, then $F1 = 0$.

27. Remark that $\sum_{j=0}^{k-1} a_{i,j}$ corresponds to all true labels that are equal to i.

2.5.4 Intersection over Union

The **Intersection over Union (IoU)**[28] is closely related to the Dice score. IoU is the area of overlap between the predicted segmentation and the ground truth divided by the area of union between the predicted segmentation and the ground truth. In the given example, the IoU score for C_1 is:

$$\text{IoU}_1 = \frac{a_{1,1}}{a_{0,1} + a_{1,1} + a_{2,1} + a_{1,0} + a_{1,2}} \qquad (\text{IoU}_1 \text{ in the table 2.2})$$

In general, the class specific Intersection over Union (IoU) for class C_i is given by:

$$\text{IoU}_i = \frac{a_{i,i}}{\sum_{j=0}^{k-1} a_{i,j} + \sum_{j=0}^{k-1} a_{j,i} - a_{i,i}} \tag{2.19}$$

Again, a macro IoU score can be calculated by taking the arithmetic mean of the class IoU scores.

Classifier accuracy

Finally, the classifier accuracy is defined as the proportion of the observations that is correcly classified:

$$\text{accuracy} = \frac{a_{0,0} + a_{1,1} + a_{2,2}}{\sum_{i=0}^{2} \sum_{j=0}^{2} a_{i,j}} \qquad (\text{table 2.2 classifier accuracy})$$

In general, the classifier accuracy is defined by

$$\text{accuracy} = \frac{\sum_{i=1}^{k-1} a_{i,i}}{\sum_{i=0}^{k-1} \sum_{j=0}^{k-1} a_{i,j}} \tag{2.20}$$

28. The IoU is also known as the Jaccard score.

Part III

Results & Experiments

Abstract

The objective of this modelling procedure is to procedure a model that can estimate segmentation masks for the 5 lumbar vertebrae from Computer Tomography (CT) and Magnetic Resonance Imaging (MRI) scans of human patients, based on point annotation of these 6 classes ($L1$ to $L5$ and 1 background class).

First, as described in chapter 2, a reference is calculated with which to compare the performance of the weakly supervised models. The reference models have the same architecture and are trained with full annotation on the same dataset as the weakly supervised models.

The first step in the modelling procedure based on point annotation masks is the construction of *single-dimension* models. Single-dimension indicates that these models investigate the performance of a model that outputs 2D segmentation masks based on 2D input images. By slicing the scan volumes along one of three central dimensional axes, a stack of two-dimensional images is created together with the point annotation maps extracted from the segmentation masks of these images. Chapter 3 presents the results of several experiments. Different model hyperparameters are tested. Models with different loss functions are trained on point annotation maps where the number of annotation points extracted from the full masks is varied.

the second step in the modelling procedure is the combination of the single-dimension models to estimate pseudo-masks. These pseudo masks can finally be used to train a model as pseudo full annotation masks. Chapter 4 first discusses the technique used to combine single-dimension models and then concludes with the results obtained by training a final model on the pseudo masks resulting from the combination.

— 1 —

Datasets

This thesis is based on five different existing publically available datasets. This chapter discusses these datasets based on different criteria:

source: reference of the owner of the dataset and brief description of the original purpose of the dataset.

Patient sample: statistics of the patients whose medical images were collected such as age, gender and possible spine pathologies.

Technical information: discusses the imaging technology, the image resolution and the spatial dimensions of the image.

1.1 Dataset overview

Table 1.1 provides an overview of the 5 datasets sourced for this work. This table shows there are important differences between the datasets in imaging mode and dataset size. Figure 1.1 also indicates the differences in image dimensions and resolutions. For more details on the data quantity, please consult chapter 1.2. Notably, the fact that some images were taken from the same patient is vital to recognize. It means the dataset is grouped.

Name	reference	imaging technology	Quantity [images]	Annotation
UWSpine	[11]	CT	125	point
xVertSeg	[13, 17]	CT	15	full
UniSiegen	[34]	MRI	17	full
PLoS	[7]	MRI	23	semantic
MyoSegmenTUM	[5]	MRI	54	full

Table 1.1: List of dataset references. The agreement with prof. T. Vrtovec regarding the xVertSeg dataset can be found in appendix C.

In table 1.2 illustrates the difference in anatomical orientation of the volumes in the different datasets. To execute the modelling approach discussed in chapter 2.1, all volume orientations were uniformized to assure that for all datasets, the volume slices along dimension 0 represent the transverse slices, the slices along dimension 1 represent the frontal slices, and the slices along dimension 2 are sagittal plane coupes of the volume.

Name	X	Y	Z
UWSpine	Left-right	Anteroposterior	Craniocaudal
xVertSeg	Left-right	Anteroposterior	Craniocaudal
UniSiegen	Anteroposterior	Craniocaudal	Left-right
PLoS	Left-right	Anteroposterior	Craniocaudal†
MyoSegmenTUM	Anteroposterior	Craniocaudal	Left-right

Table 1.2: List of dataset orientations. Clarification of the names of the anatomical axis is given in figure 1.2. The names of the axis, X, Y and Z represent the `numpy` axis `0`, `1` and `2`.
† The Craniocaudal axis in the PLoS dataset is inverted.

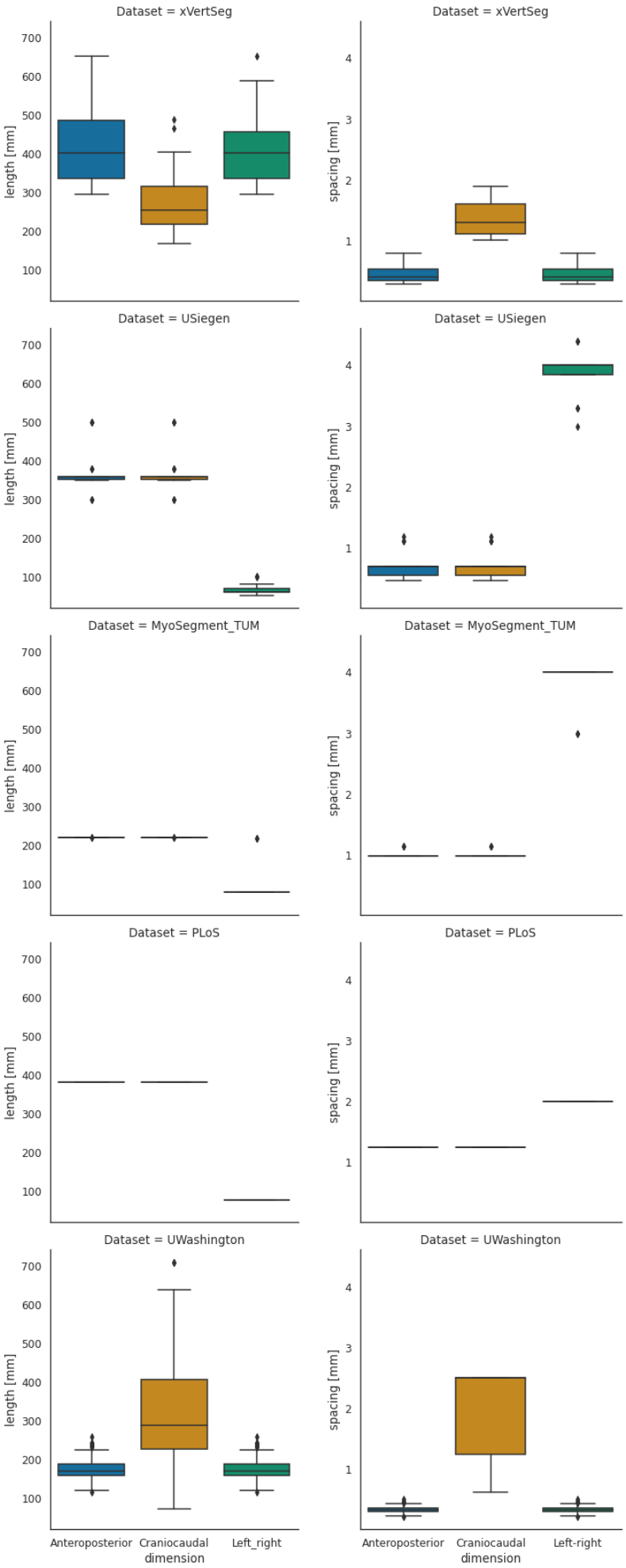


Figure 1.1: Boxplots comparing the scan dimensions and initial resolution of the different datasets. The left column indicates the volume dimensions (in mm) in the different datasets. The graphs indicate there are important differences between the datasets. Notice that both the USiegen and the MyoSegmenTUM dataset are cropped severely in the left-right direction. The right column represents the vixel spacings in the different volumes of the different datasets. In the preprocessing step all volumes are resampled on a $1mm \times 1mm \times 1mm$ grid, see page 22.

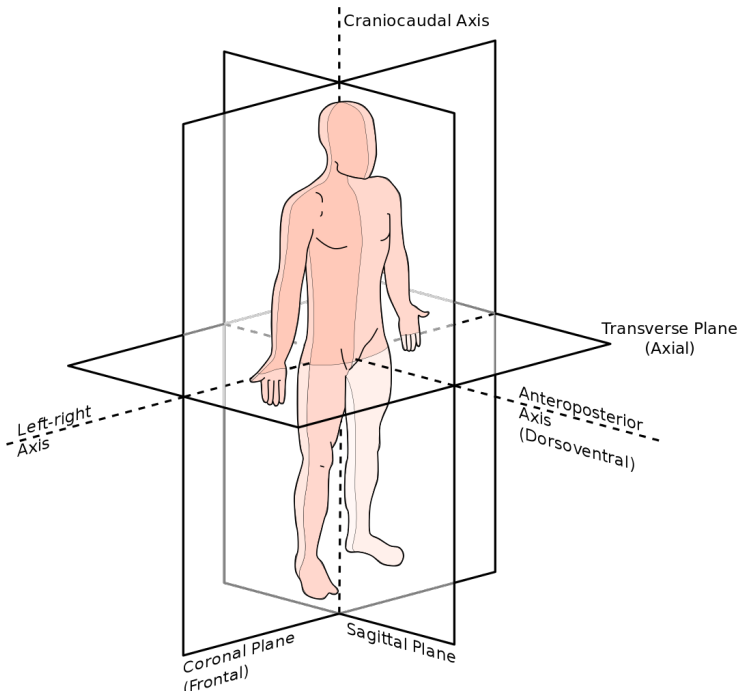


Figure 1.2: Clarification of the anatomical planes. For some images further in this document, for example figure 4.2 at page 64, the anatomical axes will be referred to by numbering: Slices perpendicular to dimension 0 are slices in the transverse plane. Slices perpendicular to dimension 1 are slices in the frontal plane. Slices perpendicular to dimension 2 are slices in the sagittal plane.

1.2 Comparison of the different datasets

1.2.1 xVertSeg

The xVertSeg [13, 17] was kindly made available by prof. T. Vrtovec (University of Ljubljana, Faculty of Electrical Engineering, Slovenia), see appendix C for the agreement. This dataset contains 25 Computer Tomography (CT) scans of the lumbar spine, of which 15 CT scans are fully labeled. Given the provided metadata[1], it is clear these 15 scans were collected from 15 different patients.

For each of these 15 scans, full instance segmentation masks for all 5 lumbar vertebrae are provided. The delineation was performed by a skilled professional.

Additionally, for each vertebra a fracture class and fracture grade is provided. Apart from vertebrae classified as *normal*, the dataset contains *mild*, *moderate* and *severe* cases of vertebrae fracture types *wedge*, *crush* and *biconcavity*.

1. patient age and gender.

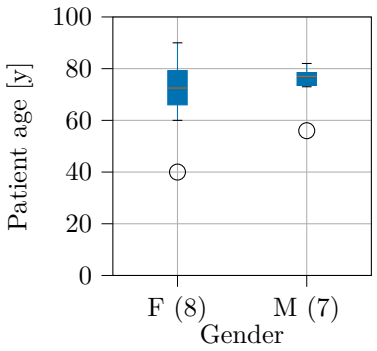


Figure 1.3: xVertSeg patients age distribution

Original Objective of the Dataset

The dataset was collected for the *xVertSeg challenge*[2]. Based on the 15 provided scans with corresponding masks, the participants were required to construct a model to make predictions on the test set of 10 unlabelled scans.

The challenge consisted of two tasks:

1. Segmentation of the lumbar vertebrae. For each scan in the test set, the segmentation masks of the lumbar vertebrae were requested.

2. Fracture classification on the segmented vertebrae, consisting of morphological grade and fracture classification.

2. see `http://lit.fe.uni-lj.si/xVertSeg/database.php`, this challenge was organized by the University of Ljubljana in 2015

Patient statistics

The patients in the xVertSeg dataset train set consist of 8 females and 7 males. The age of these patients is slightly higher than for the other datasets. The average patient in this dataset is 71 years old. A box plot of the age distribution between genders is shown in figure 1.3.

	L1	L2	L3	L4	L5	Total
biconcave	6	7	8	4	2	27
normal	5	6	4	4	7	26
wedge	3	2	2	3	2	12
crush	1	0	1	4	4	10

Table 1.3: Every patient in the xVertSeg dataset suffers from at least one spine pathology. Most of these pathologies are identified as *mild*. This table counts the spine pathologies and normal vertebrae observed over all 15 patients in the xVertSeg dataset.

Technical information

The xVertSeg dataset contains 15 CT annotated scans. On figure 1.4, two slices of the same patient (002) are represented. The slices show the complete, uncropped sections of the torso and abdominal region of the patient. There is no RoI cropping in the transverse plane. The average left-right axis span of the volumes is 417 mm. Along the anteroposterior direction, this is 421 mm. Along the craniocaudal axis, the volumes are cropped. The average volume extend in this direction is 279 mm.

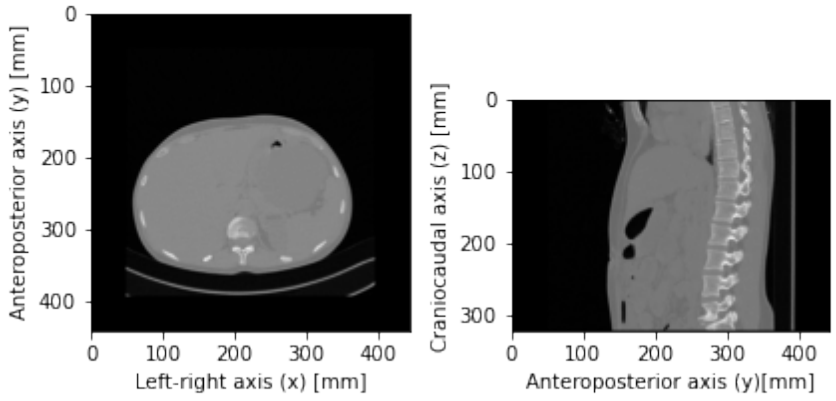


Figure 1.4: xVertSeg scan *image002*. In the xVertSeg dataset, the volume left-right axis is not cropped. These images only show part of the sacrum, but multiple thoricic vertebrae are visible.

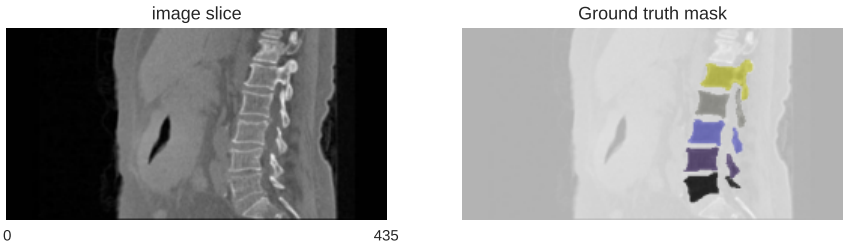


Figure 1.5: xVertSeg scan *image004*. This sagittal slice shows the xVertSeg labels include the *vertebra lamina*. Colour legend: L1 L2 L3 L4 L5

The distribution of the scan dimensions is shown in figure 1.1. This also illustrates that the *xVertSeg* dataset consists of large, high-resolution images. Mind that during pre-processing, the scans are resampled on a $1mm \times 1mm \times 1mm$ grid.

1.2.2 UniSiegen dataset

This dataset is made available in 2014 by the University of Siegen, Germany. Dr D. Zukic [34] constructed it as part of his PhD project. This dataset contains 26 MRI scans of 17 different patients[3].

3. This is not clearly stated, but can be inferred from the metadata.

The fact that scans of the same patient are correlated will be taken into account in the train, validation and test split. For more details on this split, see section 1.2 on page 23.

Original Objective of the Dataset

This dataset was collected from several hospitals (Sarajevo, Marburg, Brisbane, Schwabach, Bad Wildungen & Prague). The MRI scanner settings were varied between the scans (T1, T2, TIRM). The PhD project objective was to build a segmentation model to automate the segmentation of the lumbar vertebrae in the MRI scans to facilitate the diagnosis of several spine pathologies such as scoliosis, spondylolisthesis [4] and vertebral fractures. The final model developed by dr. D. Zukic consisted of a Viola-Jones detector for detection and vertebral body size approximation. The average Dice score compared to the manual reference was reported to be 79.3%.

4. Spondylolisthesis is the displacement of one spinal vertebra compared to another.

Patient statistics

In [34], it is not entirely made clear which scans are taken from the same patient. It is made clear, however that the 26 scans were not obtained from 26 patients. The information was inferred from the naming of the scans and the provided gender en age information[5].

Figure 1.6 illustrates that the USiegen dataset contains almost double the number of female patients compared to male patients. These patients are relatively young compared to the patients in the *xVertSeg* dataset.

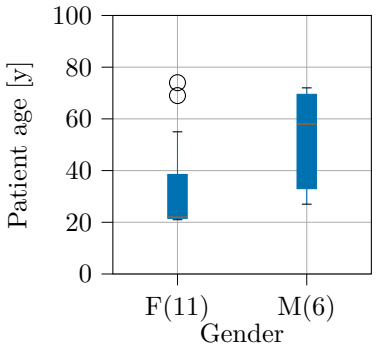


Figure 1.6: USiegen patients age distribution

Only three of the patients in this dataset were categorized as having no spinal pathologies.

Technical information

Several Magnetic Resonance Imaging techniques were used to obtain the dataset: T1, T2 & TIRM. I do not take into account this factor in the model development or the dataset split.

The volumes in the USiegen dataset are strongly cropped. Both in the anteroposterior and the craniocaudal direction, the volumes are on average 370 mm. In the left-right direction, however, the volumes have been cropped severely. The volumes are, on average, only 68 mm wide. The images have been cropped to only include the RoI. Along this left-right dimension, the voxel spacing is large.

The original scans in the *USiegen* dataset were cropped in the *left-right* direction. Although the scan resolution is relatively high in the Sagittal planes, the slice spacing along the left-right axis is coarser (see figure 1.1).

5. Wrongfully assuming two scans come from the same patient does not cause data leakage.

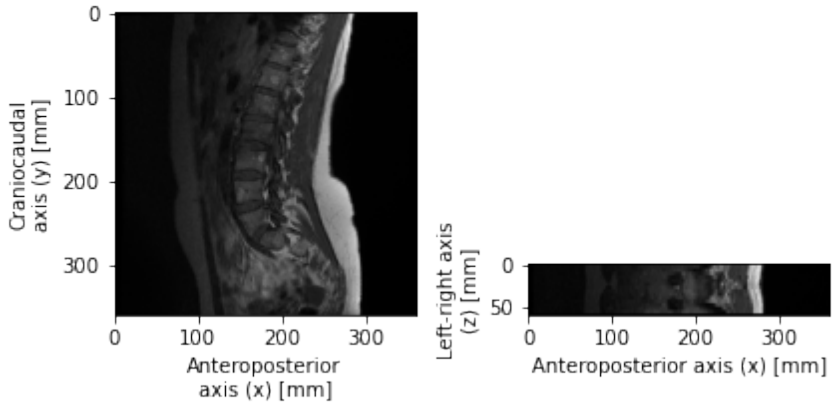


Figure 1.7: USiegen dataset scan *Aka3*. . It is immediately clear the USiegen volumes are cropped differently than the xVertSeg volumes. In the craniocaudal direction, both sacrum and coccyx are visible. Along the left-righ axis, the volumes have been cropped severely.

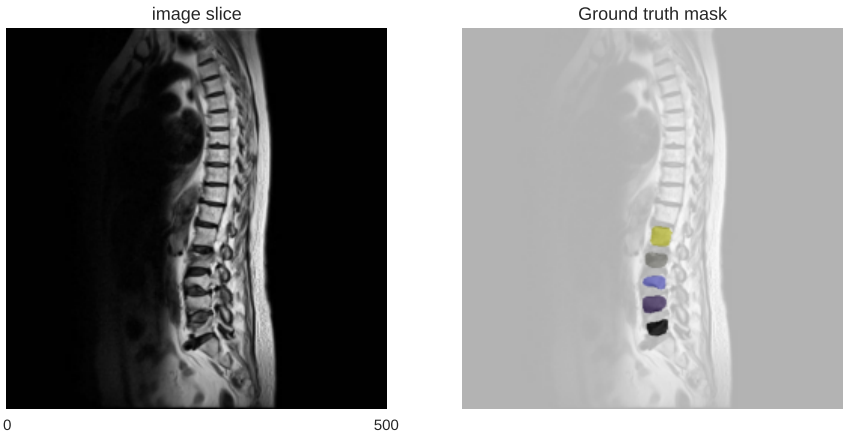


Figure 1.8: Sagittal slice of USiegen dataset scan compared with the Ground truth mask for this scan. This image illustrates a spine with a crushed vertebra. It can also be remarked that the ground truth masks provided for the USiegen dataset only contain the vertebra body, not the vertebra laminae.
Colour legend:
● L1 ● L2 ● L3 ● L4 ● L5

1.2.3 PLoS Dataset

The *PLoS* dataset was compiled for [7] in 2015 by dr. C. Chu, University of Bern, Bern, Switzerland and made publically available [6] . It consists of 23 T2-weighted spine MRI scans. Contrary to other datasets, the segmentation labels in this dataset do not distinguish the individual vertebrae from each other.

6. See : `http://doi.org/10.5281/zenodo.22304`

Original Objective of the Dataset

In [7] the development of a random forest regression approach for spine vertebrae segmentation and classification is described. The results of several random forest regressors and classifiers are unified with a voting mechanism. This approach obtains a mean Dice metric score of 88.7%.

Patient statistics

Due to the anonymization process, the *PLoS* dataset does not contain patient information. This means that it is not possible to derive any statistics regarding patient age or gender.

Technical information

As is indicated in figure 1.1, and in figure 1.9, the PLoS image volumes are cropped in the left-right direction. The volumes are consistent $381mm \times 381mm \times 78mm$, where the shortest dimension is in the left-right direction.

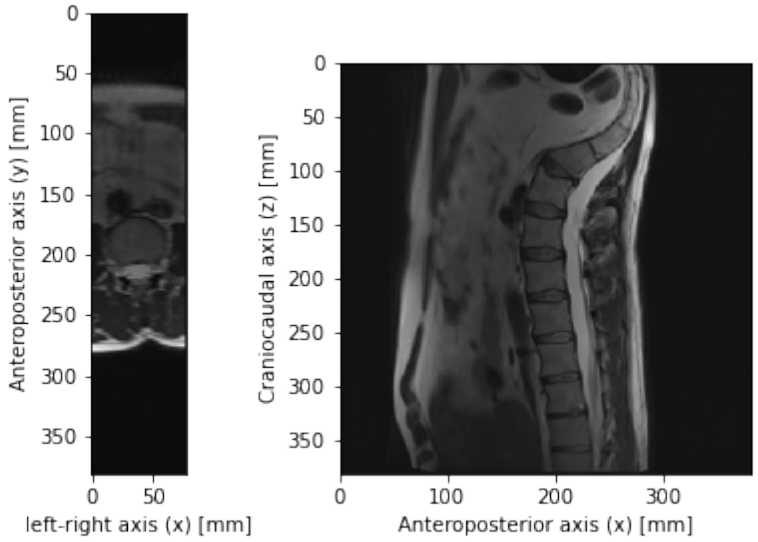


Figure 1.9: PLoS dataset scan *image002*. . The craniocaudal direction is cropped in a way comparable to the volumes in the Siegen dataset, but the direction of this axis is inverted. The left-right axis is cropped, similar to the Siegen data volumes.

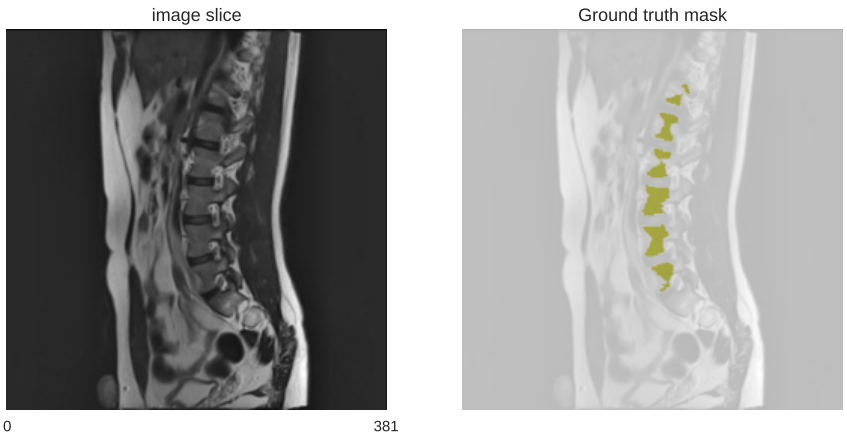


Figure 1.10: PLoS dataset scan 008 saggital slice, compared to the ground truth mask. The Glsgroundtruth masks for this dataset do not include the vertebra laminae, only the vertebra body. The mask only contains one class. No distinction between different lumbar laminae is available.
Colour legend:
● L1, L2, L3, L4 & L5

1.2.4 MyoSegmenTUM datset

This dataset is made available by S. Schläger from the Technische Universität München via the Open Science Foundation (OSF) [7]. It was constructed for the MyoSegmenTUM project [5]. It consists of 54 collections of MRI scans of the spine. In this work, only the T2 weighted Magnetic Resonance Imaging scans are used. The dataset also contains volumes with enhanced fat tissue response. Since the objective of this work is related to bone tissue rather than fat tissue, these volumes were not used. Neither was the segmentation masks for the different dorsal muscles, which are also present in this dataset.

7. see `https://osf.io/3j54b/?view_only=f5089274d4a449cda2fef1d2df0ecc56`

Original Objective of the Dataset

The MyoSegmenTUM Spine dataset is compiled as a reference dataset for developing segmentation algorithms of the lumbar spine vertebral bodies and muscle groups. Information about this project can be found in [5].

Patient statistics

In figure 1.11, the age distribution of the patients in the MyoSegmenTUM dataset is shown. There are more women (39) included in this dataset than men (15). The male patients in this dataset are, on average, clearly younger than the female patients.

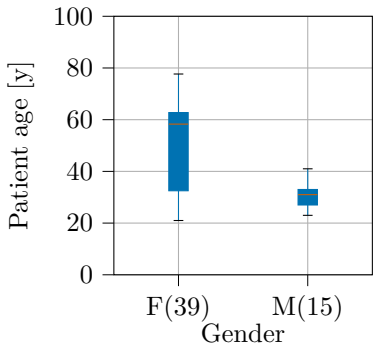


Figure 1.11: Distribution of patient age in the dataset from the MyoSegmenTUM project.

Technical information

As shown in figure 1.12, coupes from the second volume of the MyoSegmenTUM dataset are shown. As is also indicated in figure 1.1, the dimensions of the MyoSegmenTUM volumes is consistent $220mm \times 220mm \times 80mm$, where the shortest dimension is the cropped left-right axis. There are only three volumes that deviate slightly from this.

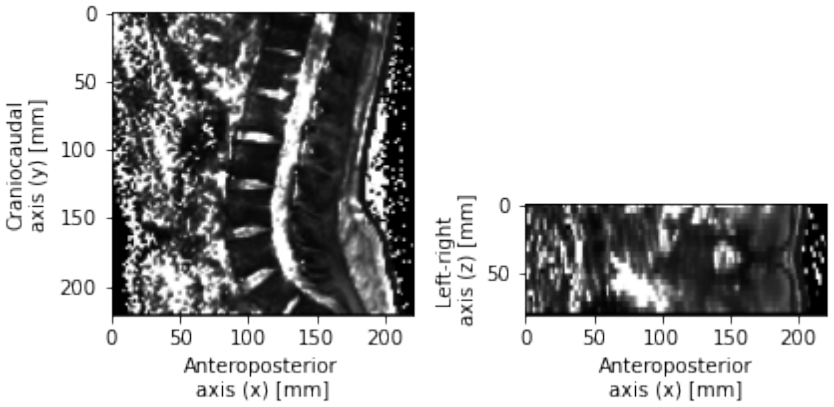


Figure 1.12: MyoSegmenTUM dataset scan *02*. The volumes are cropped in the left-right direction.

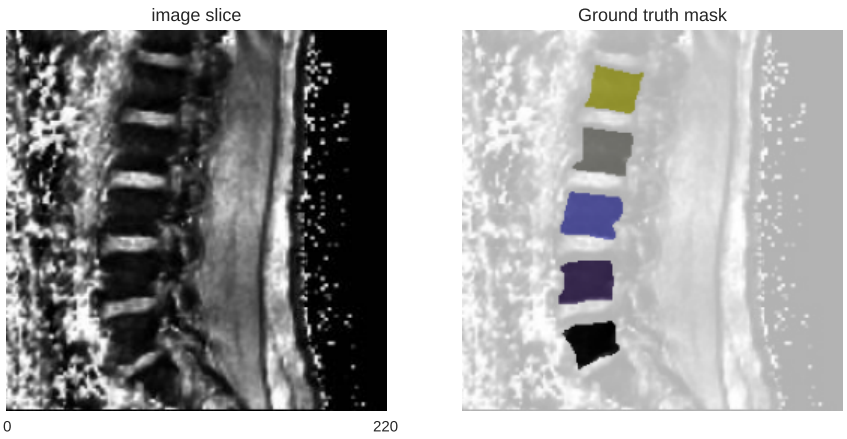


Figure 1.13: The sagittal slice of MyoSegmenTUM volume 20 is compared to the Ground truth mask. For this dataset, only the vertebra body is included in the mask.
Colour legend:
L1 L2 L3 L4 L5

Remark: For three volumes (nr 33, 53 and 54) for which the dimension of the image volume and the label mask do not correspond. It is not clear how these masks should be used. These volumes were discarded, bringing the final number of volumes used from the MyoSegmenTUM dataset to 51.

1.2.5 UWSpine dataset

This dataset is made available by the Department of Radiology of the University of Washington[8]. It has been constructed by dr. Glocker and team [10, 12] (Microsoft research) in 2012. For each scan, manual annotations of vertebrae centroids are provided. This dataset contains 242 CT scans of 150 different patients. This dataset does not contain full mask labels, only centroid point annotations. To investigate the relative performance of a weakly supervised model compared to the performance of a fully supervised model, both will be trained on the same dataset[9]. Furthermore, the evaluation of the models is based on the full annotations. Thus, the UWSpine dataset will not be used for model training. It will only be used for visual evaluation of the model on a completely new dataset[10].

8. Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, dataset available at `https://biomedia.doc.ic.ac.uk/data/spine/`

9. The modelling concept is further discussed in chapter 2.1.

10. The UWSpine dataset is *completely* new in the sense that no samples from this dataset (this *population*, so to speak) are present in the train or validation set. For the *normal* test set, other samples from the same datasets where present in the train and validation sets.

Original Objective of the Dataset

In [10, 12] the development of a model based on regression forests and a Hidden Markov Model (HMM) for vertebra localisation without needing strong assumptions on what part of the spine is visible.

Patient statistics

Figure 1.15 illustrated that the patients in the *UWSpine* dataset are relatively varied in age. Of most patients in the dataset, multiple scan images are available. The highest number of scans taken from a single patient is 5.

Technical information

Only point annotations are available for the *UWSpine* dataset. This means this dataset can only be used for weakly supervised model training.

The scans in the *UWSpine* dataset are strongly cropped around the spine, both in the left-right direction and the anteroposterior direction.

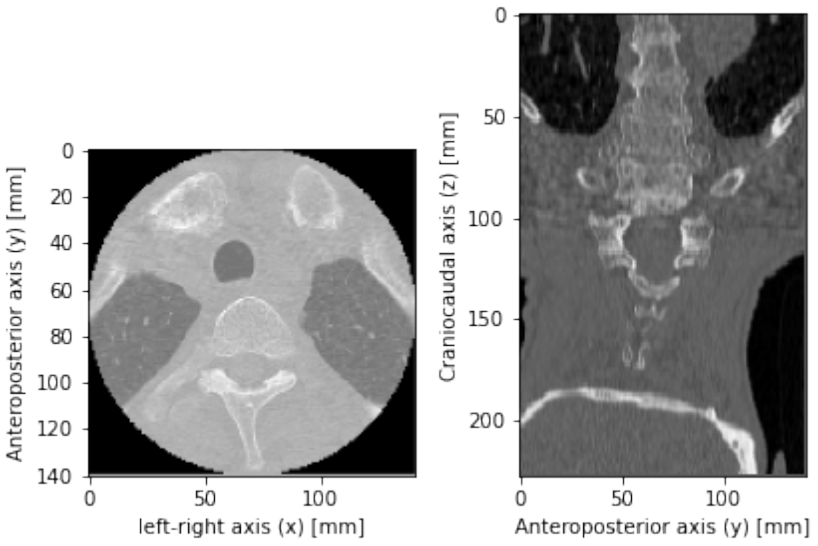


Figure 1.14: University of Washington dataset, scan *4564688*.

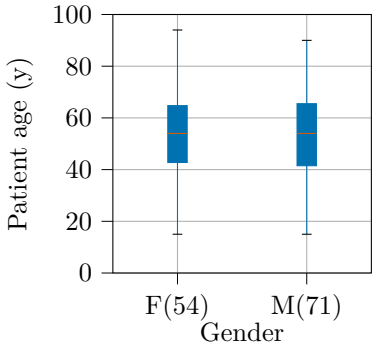


Figure 1.15: Distribution of patient age in the dataset from Washington University.

— 2 —

Reference model

As a reference to compare the models trained on weakly-supervised data with, the model performance of a model trained on the same fully supervised data is taken as a reference. In this chapter, the results of these fully supervised reference experiments are discussed. The metric based on which the experiment results are compared is the inverse class weighted dice score, see equation 2.18 on page 42.

values in [%]		Predicted					
		BG	**L1**	**L2**	**L3**	**L4**	**L5**
Actual	**BG**	99.9	13.2	14.5	14.2	12.9	12.1
	L1	0	86.6	2.1	0.1	0.2	0
	L2	0	0.2	83.4	0.9	0.1	0
	L3	0	0	0.1	83.9	0.4	0
	L4	0	0	0	1	86.1	0.7
	L5	0	0	0	0	0.3	87.2

Table 2.1: Confusion matrix for the model trained with full label masks (network VGG16-FCN8 and cross-correlation loss), evaluated on the test set. The values have been normalized by the total number of voxels predicted in each class. The diagonal elements are thus the class precisions: $\mathcal{P}(C = L1 \mid pred = L1) = 0.866$ while $\mathcal{P}(C = L2 \mid pred = L1) = 0.132$.

2.1 Experiment results

The fully supervised experiments serve two goals. The first is to calculate a reference model performance with which to compare the weakly supervised model results. The second is to support hyperparameter choices for the point supervised experiments. It is assumed that a network architecture that yields a good, fully supervised model is also a suitable choice to build a weakly supervised model. The possible influence of the context slices[1] is also evaluated with these experiments.

1. The context slice idea is discussed in detail in chapter 2.3.2 on page 33.

In figure 2.1, the results of the reference experiments are shown. These results show the network based on VGG16 yields better results than the alternatives based on RESNET50 and U-Net. Despite what was hoped for, the context slices do not seem to increase the model performance. Remarkably, there is little difference between the models trained with a weighted cross-entropy loss and the non-weighted cross-entropy loss[2] when considering the weighted dice score. In figure 2.2, the model result based on the FCN8 VGG16 model without context slices is compared in detail for the models trained with weighted and the unweighted cross-entropy function. Based on these images, some conclusions can be drawn:

2. The weighted cross-entropy loss is defined in chapter 2.4.1 on page 39. The objective of a weighted cross-entropy is to improve the model performance for under-represented classes in an unbalanced dataset by adding a factor inverse proportional to the class prevalence to the cross-entropy loss.

1. The prediction of the background class is practically perfect for all

datasets.

2. Other authors [24, 8] required elaborate evaluation schemes to be able to label the segmented lumbar vertebrae. Due to the larger view ($352mm \times 352mm$ vs $180mm \times 180mm \times 180mm$) this model can use by working with 2D slices, it is possible to have all vertebrae in one image. The latter proves to allow the model to be trained to identify all five lumbar vertebrae without requiring the evaluation scheme. Observe that $\mathcal{P}(C = Li \mid pred = Lj)$ is low for $i \neq j$.

	precision	**recall**	**dice**	**iou**
Background	99.9%	99.7%	99.8%	99.6%
L1	55.8%	84.0%	67.1%	50.5%
L2	70.2%	85.2%	77.0%	62.6%
L3	77.8%	79.9%	78.8%	65.1%
L4	74.0%	83.2%	78.3%	64.4%
L5	72.8%	88.2%	79.7%	66.3%

Table 2.2: Per class metric results for the reference model. The performance metrics on the background class are invariably excellent. This is caused by the unbalance in the data. Using the inversely weighted dice score gives more weight to the underrepresented classes, which are more of interest.

The model trained with an unweighted cross-entropy loss underperforms compared to the model trained with the weighted cross-entropy loss to predict the $L1$ class. For the other classes, the conclusion based on the inverse weighted dice score holds.

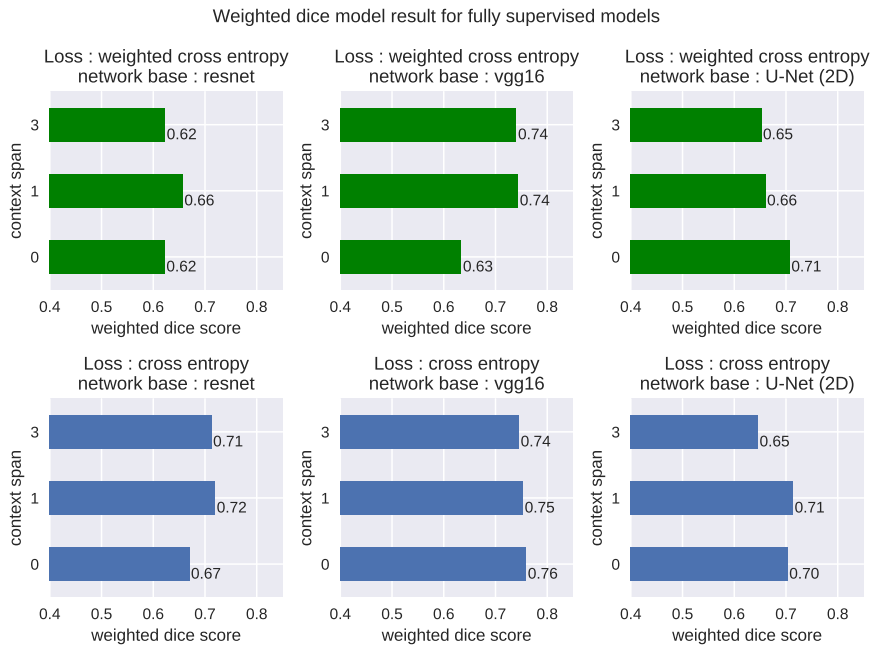
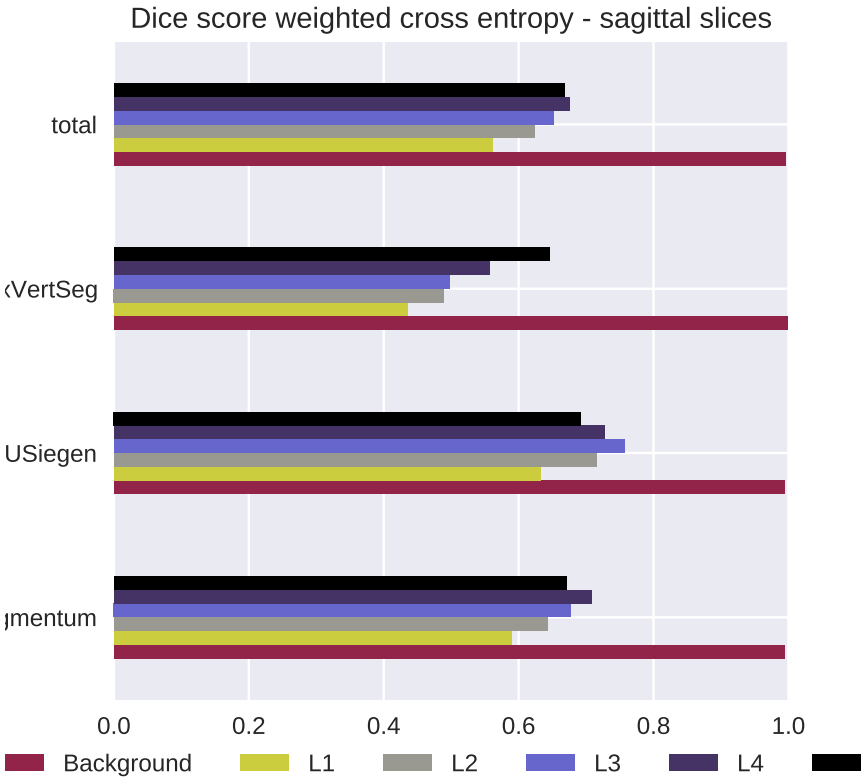


Figure 2.1: Results of the fully supervised experiments. The indicated model performance metrics are calculated on the test set. The columns represent de weighted dice scores for models based on the same network architecture with different loss functions.

2.2 Conclusion

Based on the results shown in figure 2.1, it was decided to perform the weakly supervised experiments with the model based on VGG16 FCN8 with one context slice. The reference performance metric to compare experiments with models trained on weakly supervised data is $\text{Dice}_{wi} = 0,76$.



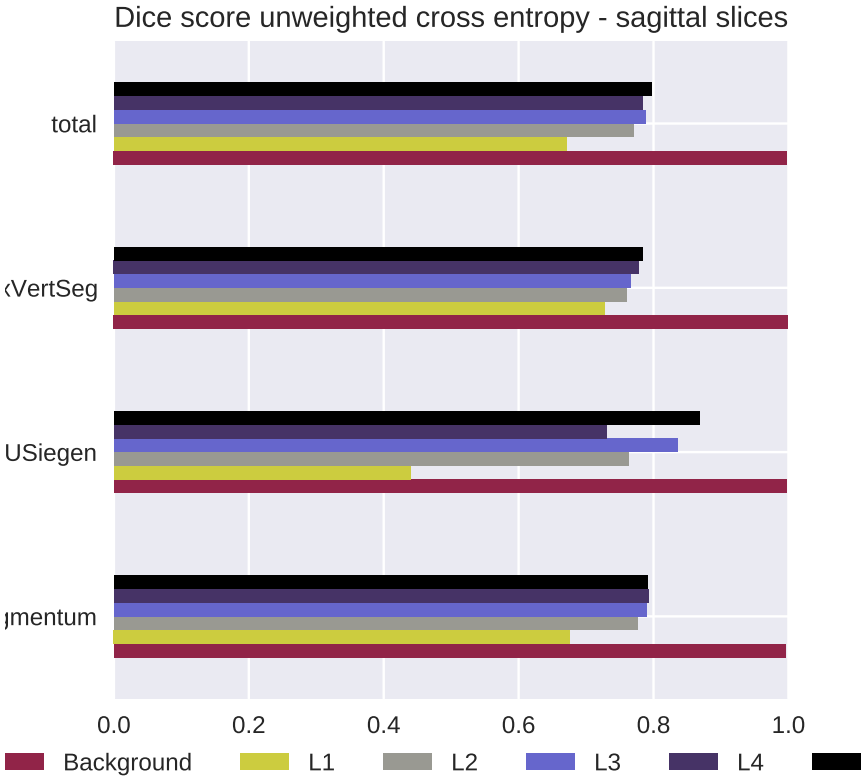


Figure 2.2: Detailed result for the fully supervised model trained without context slices and with a weighted cross entropy loss function. The model trained with the unweighted loss performs better than the result shown in figure 2.2 for the weighted cross entropy loss function.

— 3 —

Single dimension experiments

This chapter discusses the results of several experiments in the construction of single-dimensional models. The objective of these experiments is to investigate the influence different model hyperparameters have on the model performance. Based on this investigation, the hyperparameters for the single-dimensional models in the final step can be chosen.

3.1 Single dimension weakly supervised models

In this section, the results of experiments with different model hyperparameters are compared to each other. The point annotation sets are generated from the available full annotation masks for each slice. In this work, different datasets were used with different levels of provided annotation. Annotation differences between different datasets are listed in table 1.1 on page 45.

For three datasets[1], full annotation masks are available, meaning that these scan volumes are labelled with volumes for each of the five lumbar vertebrae. For each voxel, the ground truth indicates whether it belongs to a lumbar vertebra and which vertebra. For one dataset (PLoS), only semantic segmentation is available. It means that the lumbar vertebrae are indicated as one class but not distinguished. With each voxel in the PLoS dataset, ground truth is associated that indicates if it belongs to a lumbar vertebra. There is no label, however, to indicate which of the lumbar vertebrae this is.

When a scan is sliced along the craniocaudal axis[2] a human can distinguish all five lumbar vertebrae (after some practice). Therefore, one may hope the single-dimension models trained on sagittal or frontal slices will be able to do the same. Even though delineating a vertebra on a transverse slice is possible, identifying which vertebra is presented is challenging for a human. Supported by a brief test, it was confirmed that trying to estimate five vertebra classes from the transverse slices only provides very confusing results. The models trained on the transverse slices only intend to label the slice pixels as either background (0) or vertebra (1), whereas the models trained on the sagittal and frontal slices intend to indicate which of 6 classes[3] the pixel belongs to.

The models trained to segment sagittal and frontal slices were trained on datasets xVertSeg, UniSiegen and MyoSegmenTUM. From the available full masks, point annotation labels were extracted. The models trained to segment transverse slices[4] are trained on the same three datasets and additionally on the PLoS dataset.

The model performance results are compared based on the weighted dice score calculated on the test set. This metric is described in chapter 2.5.3 on page 42.

1. The three datasets for which complete annotation is available are the xVertSeg dataset, the UniSiegen dataset and the MyoSegmentum dataset. Of which the MyoSegmentum dataset is the largest by far.

2. The nomenclature of anatomical planes and axis is illustrated in figure 1.2 on page 48. The craniocaudal axis is the vertical axis when the patient is standing up.

3. 0 for background and $i \in [1, 2, 3, 4, 5]$ for the corresponding $Li \in [L1, L2, L3, L4, L5]$.

4. This segmentation is, as stated not based on distinguishing separate lumbar vertebrae from each other.

3.1.1 Evaluation of the model Hyperparameters

Several tests were conducted to estimate the influence of model components and model hyperparameters[5] on the model performance.

5. One could argue that the number of annotation points is not a *model* hyperparameter. It is an essential parameter for the *modelling* approach in general.

Weighted vs unweighted point loss performance

Equation 2.5 on page 39 defines the point loss $\mathcal{L}_P$. In this equation, the factor $w_{\mathcal{Y}_i(\vec{p})}$ indicates a weight that can be assigned to each of the output classes. Since data classes can be unbalanced, these weights can help to counter this imbalance. In this problem (based on the available datasets), there are about 500 times more background voxels than voxels belonging to a lumbar vertebra. By weighing with a factor proportional[6] to this ratio, this imbalance can be countered[7]. In the unweighted case, $w_\cdot = 1$.

6. The weighting vector is normalized.

7. When full data labels are available, the counts of the voxel types are available for the train set (one should not include the counts of the validation & test set to avoid data leakage). In principle, this information is not necessarily available when only point level annotation is available. In this work, it is considered that (at least an approximation) of the ratios can be available as prior knowledge.

In figure 3.2, the difference between the weighted dice score & the average dice score on the test set is compared for two models trained on the transverse slices. This result shows that the result of the non-weighted segmentation is better than the result of the model trained with weighted loss. At first, this seemed surprising. Contrary to fully supervised models, in the weakly supervised case, the ratio between *labelled* points is not as unbalanced as the ratio between the actual class pixels in the result. This explains the weighted point loss performance compared to the unweighted point loss performance. An approach that is not tested in this work is to weigh proportional to the inverse of the number of *annotation* points per class instead of weighing proportional to the number of *ground truth* pixels[8]. The disappointing performance of the model trained with a weighted point loss compared to the model trained with the unweighted point loss was investigated deeper. Due to the overweighing of the classes, the model trained with the weighted point loss does not predict the background class anymore! This results in a recall score of 0.00 for this class. Based on these observations, the subsequent models in this work are trained with the non-weighted point loss component. As mentioned earlier, one could investigate other weighting schemes to optimize concerning the point labels provided. Even though this is an interesting topic, it was not investigated further in this work.

8. It seems worthwhile for future researchers to investigate such an approach deeper. As it is clear from this chapter and chapter 3.1.2, extracting more annotation points (or asking the expert to provide more annotation points) does not necessarily guarantee a better model performance.

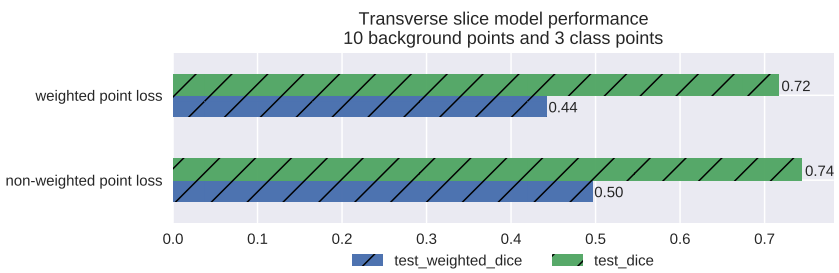


Figure 3.1: Illustration of the difference in model performance (inverse class weighted dice score) between a weighted point loss function and the unweighted point loss function for point annotated models trained on transverse slices.

Figure 3.1 shows the result of models trained on the same datasets of transverse slices. These models are trained with point annotations extracted from the available ground truth masks. For these models, 10 points were extracted from the background mask, and 3 points from each instance of each class in the slice. The texture on the plots of the transverse model performance bars tries to help the reader identify that these models do not distinguish the vertebrae from each other but only predict a voxel to be a vertebra

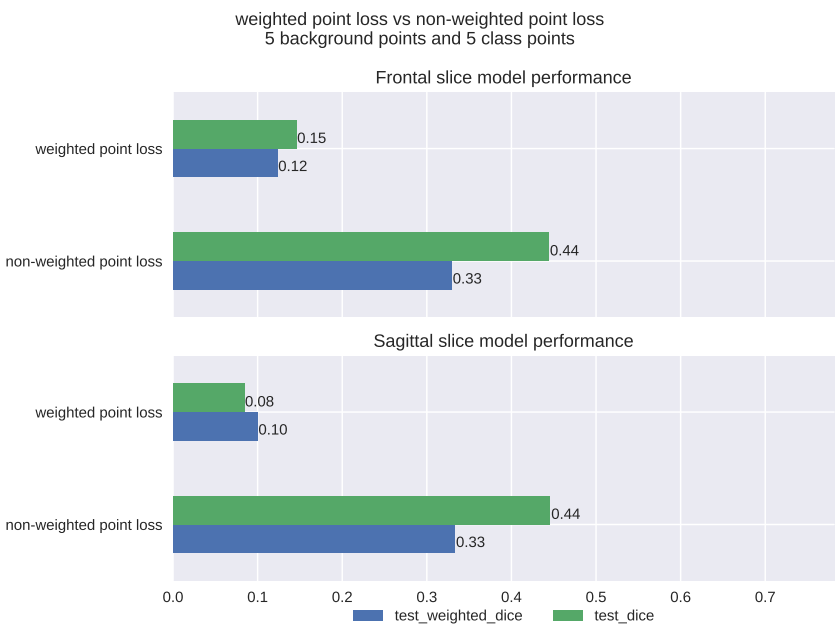


Figure 3.2: Illustration of the difference in model performance between a weighted point loss function and the unweighted point loss function for point annotated models trained on frontal slices. The models are trained with point annotations extracted from the available ground truth masks. For these models, 5 points were extracted from the background mask and 5 points were extracted from each instance of each class in the slice. The bar in blue indicates the inverse weighted dice score, evaluated on the test set. The bar in green indicates the average dice score over the classes.

or not. Figure 3.2 shows the result for transverse slices. Remark that the number of class annotation points and background points is different between figure 3.1 and figure 3.2.

Value of the added loss components

In this work, two-loss components were added to the consistency loss published in [22]. Where [22] is based only on the point loss $\mathcal{L}_P$, and the consistency loss $\mathcal{L}_C$, this work makes use of two extra loss components: the prior extend loss $\mathcal{L}_E$ and the separation loss $\mathcal{L}_S$. The four-loss components used in this work are described in more detail in 2.4. Figure 3.3 shows experimental results validating the positive influence these added loss terms have on the model performance.

3.1.2 Evolution of the model performance with increased labelling effort

The most basic modelling hyperparameter for a point annotation modelling campaign is the number of labelling points one asks the expert to provide. This section presents experimental results to estimate the number of annotation points on the resulting model performance. Intuitively, one would expect the model performance to increase with the number of annotation points provided. This hypothesis turns out to be false.

Figure 3.3 shows the weighted dice score decreases with more annotation points for models trained with unweighted point loss. The model becomes very eager to find class points. This causes many background pixels to be wrongly classified as class points. The resulting decrease in precision causes the F1 score to decrease. It might be interesting for future researchers to investigate the influence of relative quantity of class and background annotation points. The time (and computational power) to investigate this behaviour completely was not available in this thesis project.

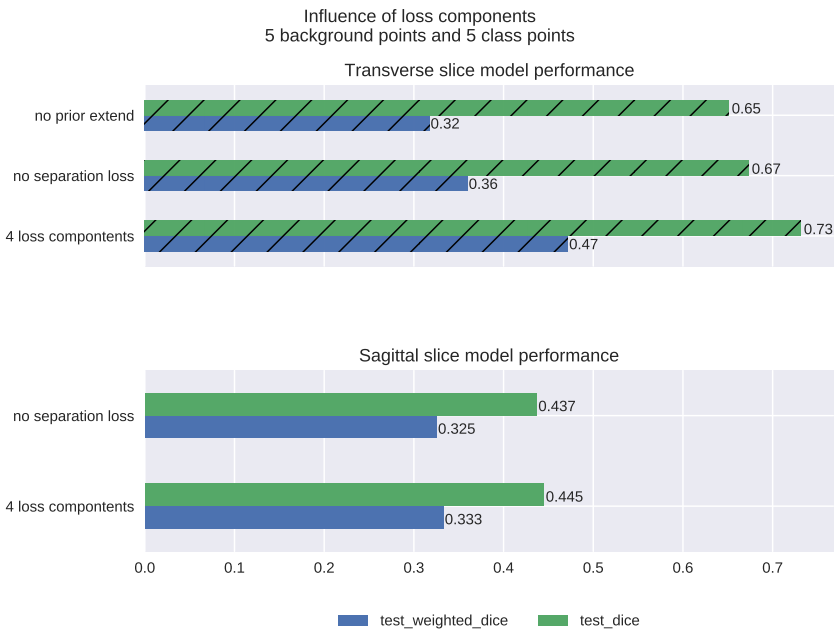


Figure 3.3: Evaluation of the added value of loss components. The bar in blue indicates the inverse weighted dice score, evaluated on the test set. The bar in green indicates the average dice score over the classes. The texture on the bars representing the transverse model performance help the reader remember the transverse model does not distinguish between the lumbar vertebrae.

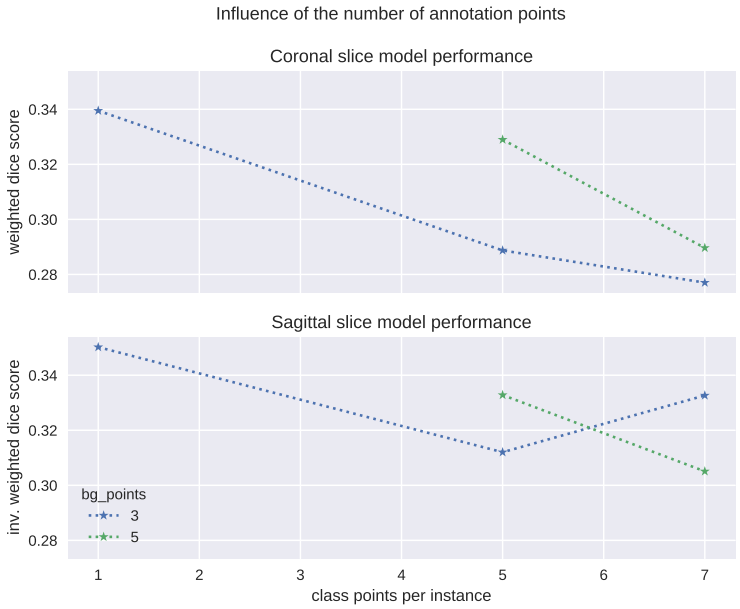


Figure 3.4: Evolution of the weakly supervised model performance with increased number of annotation points on the inverse class weighted dice score of the point annotated models. Eventhough increasing the number of class annotation points does not improve the model performance, there seems to be an increase in model performance possible by increasing the number of background points.

— 4 —

Single dimension model combination

In chapter 3, the construction of single dimension models is discussed. These models extimate the segmentation masks of $352mm \times 352mm$ patches of scan volume slices along one of the three main axis. In this chapter, the results of these single dimension models are combined to form a *pseudo* mask that is subsequently used as labelling to train the final segmentation network. Details regarding this combination procedure are given on page 30. Interesting to note in this procedure is that the evaluation metric of these obtained segmentation masks is improved in each of these steps. The pseudo mask volume performs better than the single dimension model results and the final model trained on the pseudo masks performs better than the pseudo mask itself.

Slice direction	**Transverse**	**Coronal**	**Sagittal**
Context Slices [mm]	1	1	1
Points per class instance	1	1	1
Background points	5	3	3
Dataset	PLoS xVertSeg USiegen MyoSegmenTUM	xVertSeg USiegen MyoSegmenTUM	
Segmentation classes	2	6	6
score $dice_{wi}$	0.46	0.38	0.38
score $dice_{wi}$ combination	0.51		

Table 4.1: Combination of three point supervised models with algorithm 1. These models were constructed with a fixed number of background points and a fixed number of class labels per class instance. This test indicates that the segmentation mask obtained from the result of single dimension models with algorithm 1 allows to obtain a new segmentation mask with a higher metric score, the pseudo masks. The inverse weighted dice scores are evaluated on the cross validation set, this causes the difference with the values in figure 3.4.

In table 4.1, the segmentation quality is calculated on the validation set. Since there are two hyperparameters (the number of denoise iterations and the number of erosion iterations) to optimise in the combination algorithm 1, the algorithm is evaluated with a matrix of these hyperparameters to choose the optimal hyperparameter set to construct the pseudo masks for the test set. In figure 4.1, this obtimization procedure is illustrated. The optimal hyperparameter set in this case was 2 denoise iterations and 1 erosion iteration.

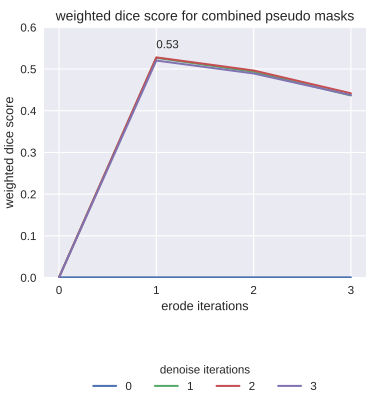


Figure 4.1: Illustration of the hyperparameter optimization procedure for the combination detailed in 4.1

The result of this procedure is illustrated in figures 4.2, 4.3 and 4.4. These images illustrate that the result of the combination procedure is an improvement compared to the results of the individual models.

The procedure described in the last two chapters now allows to train a network with only point annotated labels by using the results of the pseudo mask volume. Mind however that the results in chapter 3 and the results combined in this chapter were obtained with a different set of point annotations for each dimension. This was done to make comparison possible. The next chapter discusses a more realistic scenario.

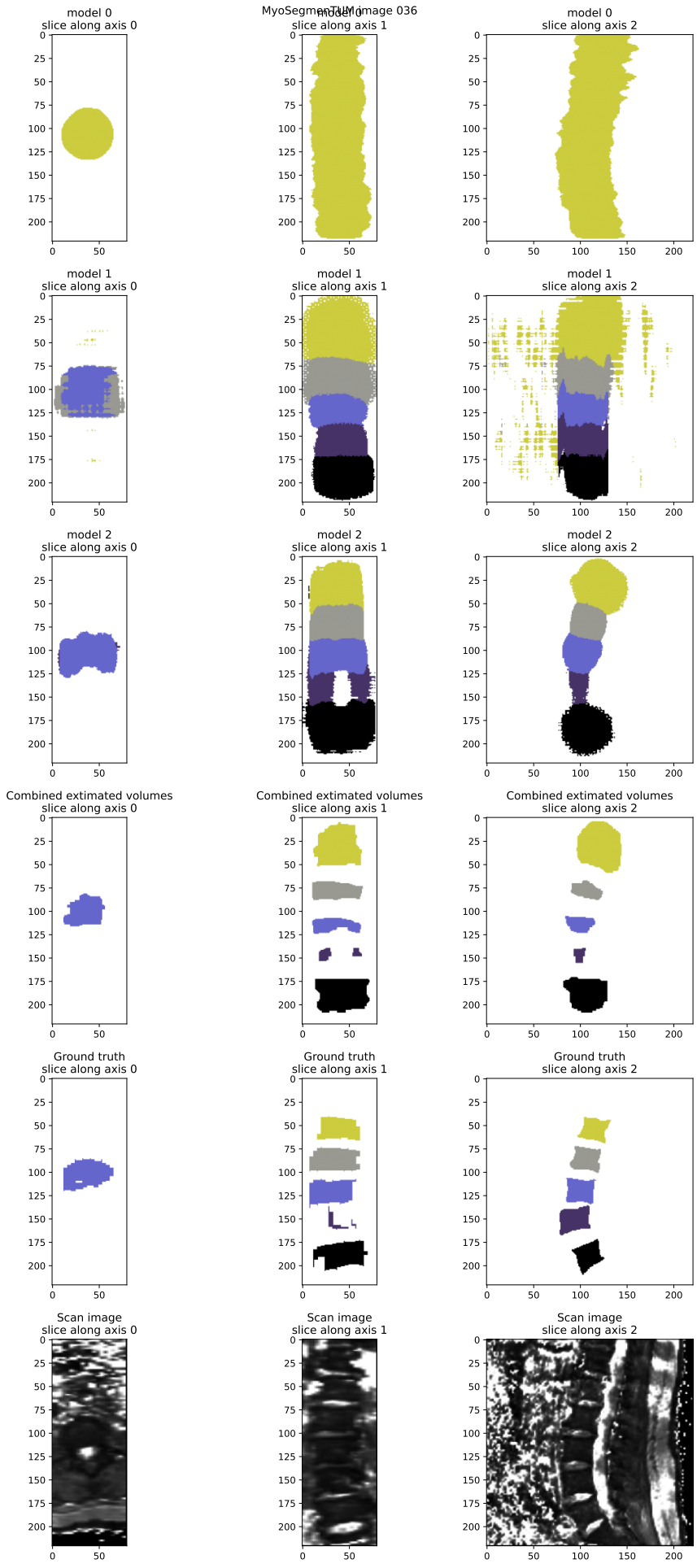


Figure 4.2: Result of the combination of the three single dimension model results for volume MyoSegmenTUM nr 36. The colours indicate the vertebra classes. Only one semantic class is estimated in the first row, illustrating the model trained on transversal slices. On the first three rows, slices of the resulting segmentations from the single dimension models are shown. It is clear these masks contain some artefacts and are not always in agreement with each other. On the fourth row, the result after mask combination and morphological smoothing is shown. This corresponds more closely to the ground truth mask, shown on the fifth row. This final mask, shown on the fourth row, will be used as a pseudo mask to approximate the unknown ground truth mask. In the last row, the corresponding images are shown.
Colour legend:
● L1 ● L2 ● L3 ● L4 ● L5

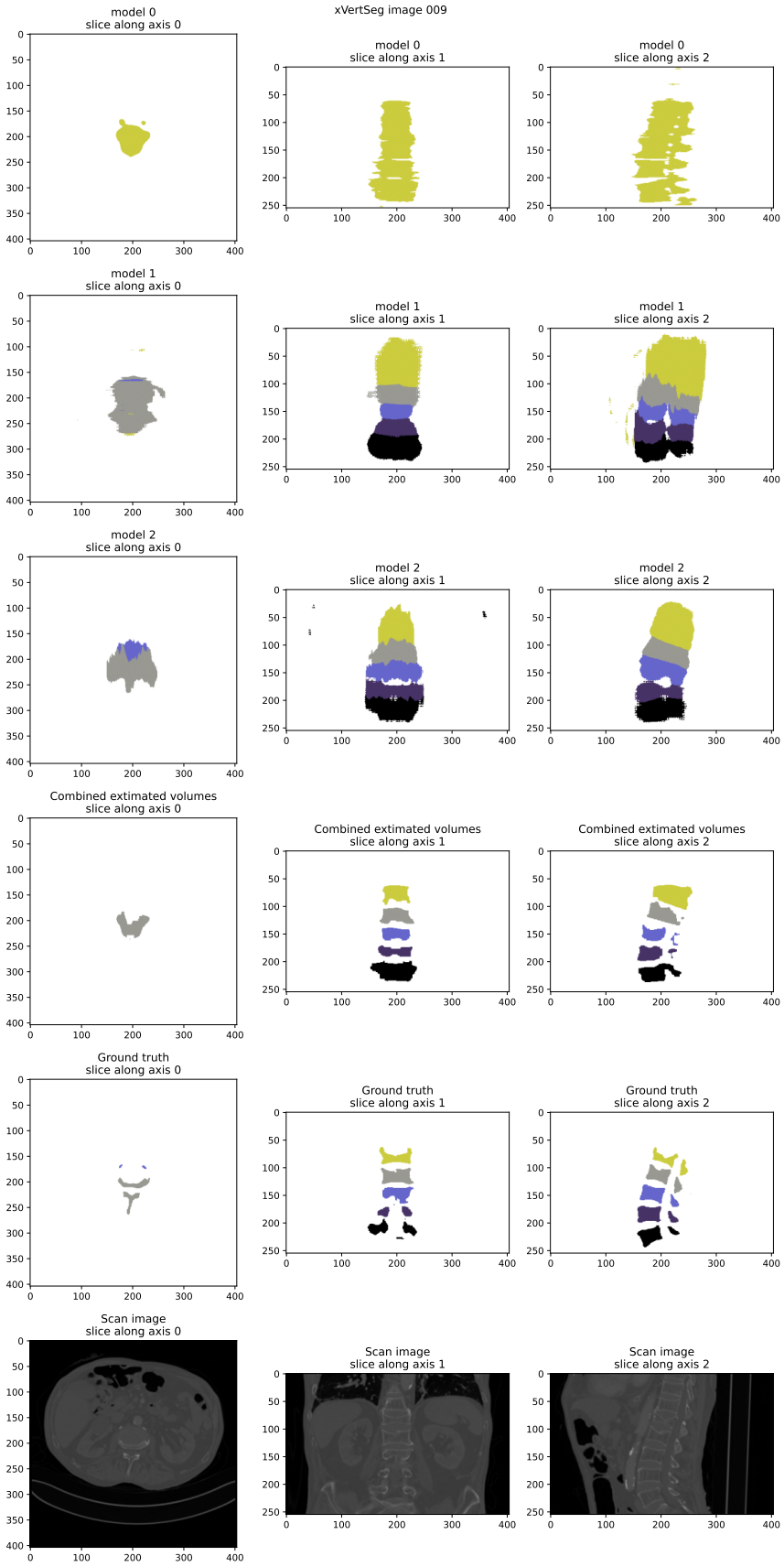


Figure 4.3: Result of the combination of the three single dimension model results for volume xVertSeg 009. The layout of this picture is identical to figure 4.2.
Colour legend:
● L1 ● L2 ● L3 ● L4 ● L5

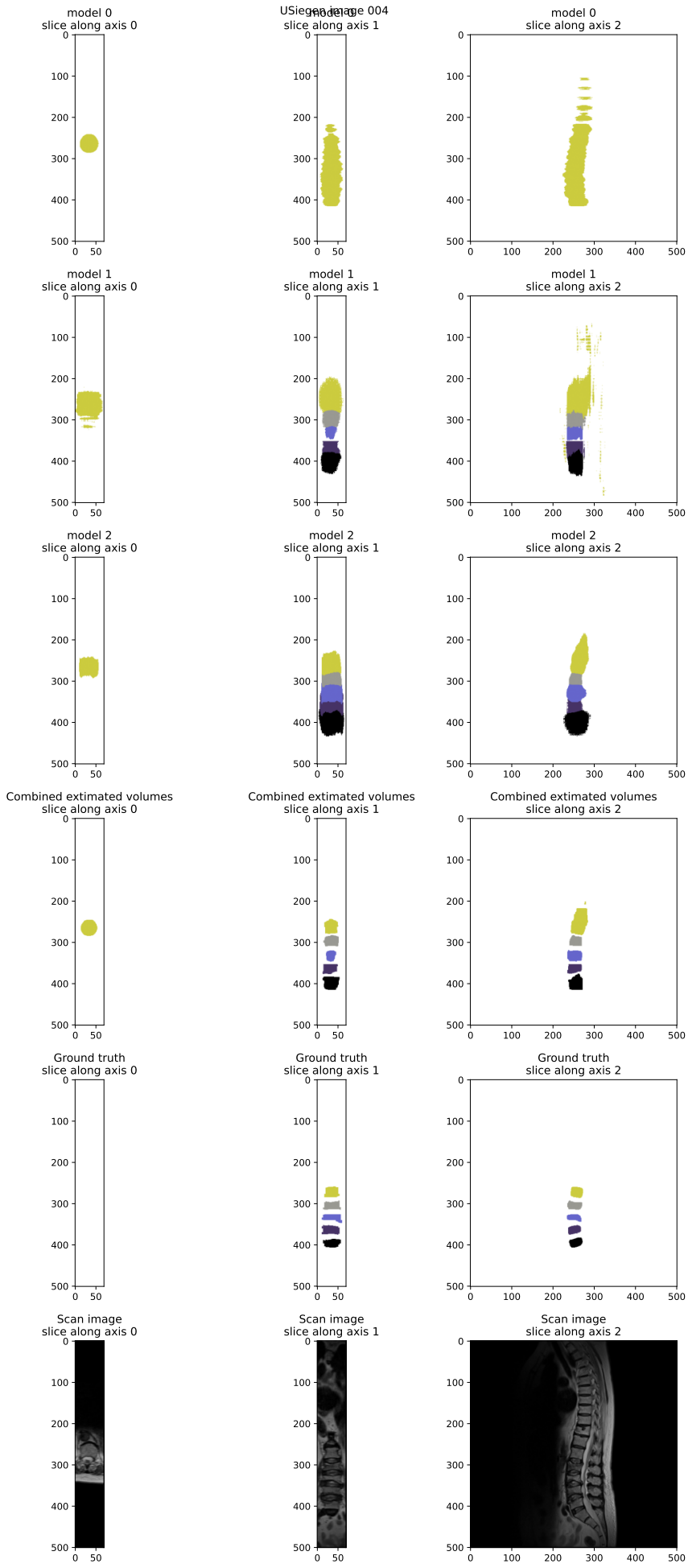


Figure 4.4: Result of the combination of the three single dimension model results for volume USiegen 004. The layout of this picture is identical to figure 4.2.
Colour legend:
L1 L2 L3 L4 L5

— 5 —

Pseudo mask training

In reality, an expert will generate a single stack of annotations, not three. As described in chapter 2.3.3 on page 33, one of the single dimensional models will be trained on this stack of annotated slices. The other two however, will be trained on stacks where the same annotation points are used, but now as these are found back when slicing in the two other dimensions. This means that for the slices along one of the geometric dimensions, a known number of annotation points will be provided. For the slices perpendicular to the other two geometric axis, the annotation points will have to be deduced from this original stack. Figures 5.2 and 5.3 illustrate this.

Inferring annotation points from labels provided for slices taken in another direction implies the number of annotation points will vary. For some slices, no annotation points will be available. As a first test, point annotations in the sagittal slices with 1 class point per class instance and 1 background point were created. This showed to decrease the number of labelled strongly. When training with these slices, the segmentation networks showed to perform very poorly (inverse weighted dice scores of 0.22, 0.23 and 0.33 for the models trained respectively on the transverse, frontal and sagittal slice stacks). As is discussed on page 11 the cost of additional data points when the first point is provided is very low. When generating 3 labels per class instance and 5 background labels, the estimated cost is only increased to about $\frac{22,1+6.0,9}{239.7} = 11,5\%$ of the cost of a full mask label. It is clear that this type of reasoning cannot be stretched to extremes (100 points or such), yet, the numbers of 3 and 5 seem perfectly reasonable.

Table 5.2 illustrates how the performance of the combination of models trained on different slice stacks results in pseudo masks which prove to be an improvement compared to the individual models. It is remarkable how high the performance of the models trained on *inferred* annotation points proves to be. Seemingly, randomizing the number of annotation points could have a positive influence on the model performance.

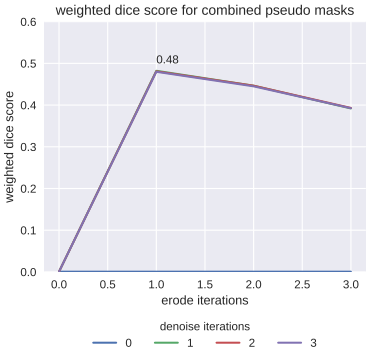


Figure 5.1: Illustration of the hyperparameter optimization procedure (see also table 5.2)

labelled slices	**1 class pt 1 BG pt**	**3 class pt 5 BG pt**
transv.	4466	9121
frontal	2627	4507
sagittal	7939	7939

Table 5.1: Available labelled slices from labelling the sagittal slice stack.

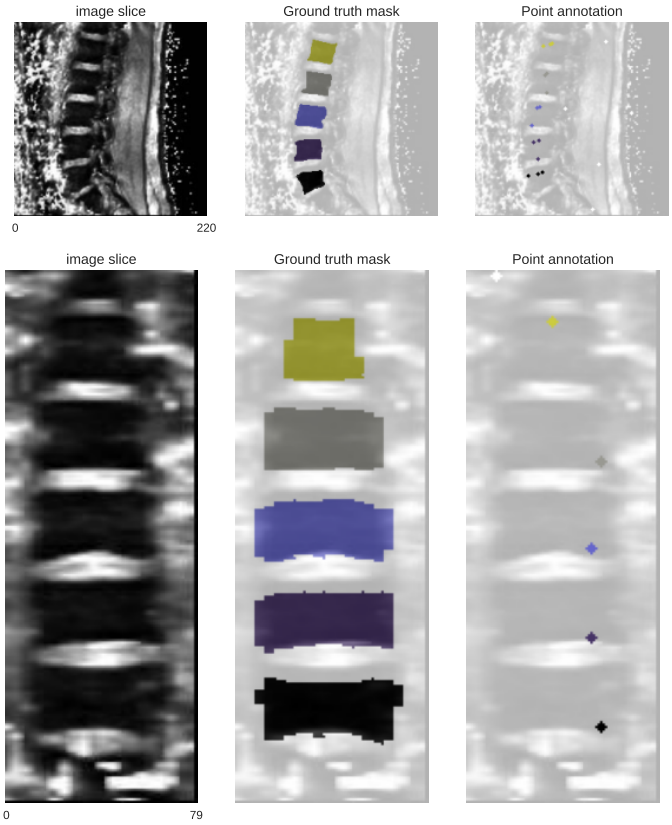


Figure 5.2: Illustration of the single-stack labelling procedure. Since the labeller is only providing point annotations for the sagittal slices, slice 21 of MyoSegmenTUM 20 is labelled with the definded 3 class labels per class instance and 5 background point labels. The same annotation points show up in the frontal slice 75 of the same volume.
Colour legend:
● L1 ● L2 ● L3 ● L4 ● L5

Slice direction	**Transverse**	**Coronal**	**Sagittal**
Context Slices [mm]	1	1	1
Points per class instance	variable	variable	3
Background points	variable	variable	5
Dataset	PLoS xVertSeg USiegen MyoSegmenTUM	xVertSeg USiegen MyoSegmenTUM	
Segmentation classes	2	6	6
score $dice_{wi}$	0.51	0.41	0.34
score $dice_{wi}$ combination	0.48		

Table 5.2: Combination of three point supervised models with algorithm 1. These models were constructed from points labels that were provided on the sagittal slice stack. The labels for the other two slice stacks were derived from this first one. This test indicates that the segmentation mask obtained from the result of single dimension models with algorithm 1 allows to obtain a new segmentation mask with a higher metric score, the pseudo masks. The weighted dice scores are evaluated on the cross validation set, this causes the difference with the values in figure 3.4 where the values were obtained by evaluation on the test set.

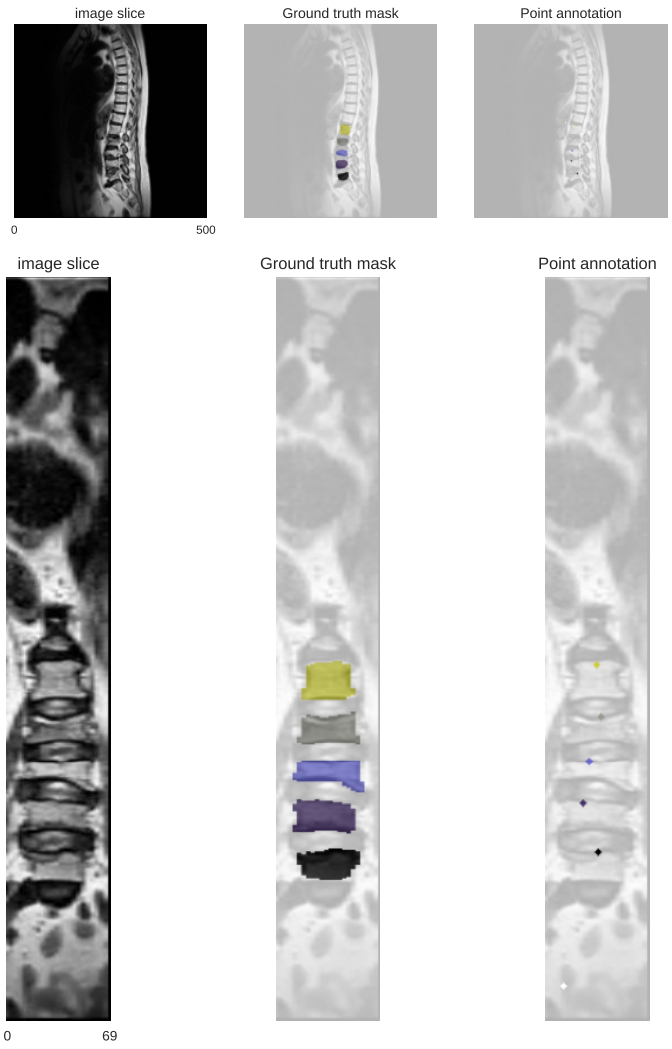


Figure 5.3: Illustration of the single-stack labelling procedure. Since the labeller is only providing point annotations for the sagittal slices, slice 20 of USiegen 04 is labelled with the definded 3 class labels per class instance and 5 background point labels. The same annotation points show up in the frontal slice 250 of the same volume.
Colour legend:
● L1 ● L2 ● L3 ● L4 ● L5

Illustration 5.4 shows the final result of the procedure. With the pseudo masks obtained from combining the three model results, a new model is trained. This last model is only trained on the sagittal slice stack. The final results obtained with this model are illustrated in figure 5.4. The hyperparameters of this model were optimized in the same way as the reference model. The unweighted cross entropy loss proves to yield better results than the weighted cross entropy loss. For the reference model, the same result was observed.

The confusion matrix of the model trained with pseudo mask labels (table 5.3) shows the model performance is clearly inferiour to the model performance of the fully supervised model. Mind however that this model is trained based on data of which the labelling only costs about 12% of the labelling cost of the fully supervised model for the train set scans. The total labelling cost will be higher since the procedure requires full labels for the model evaluation in the cross validation and the test set. If one takes, is was done in this thesis, both the cross validation and test set to be $\frac{1}{6}$ of the total volumes, the total cost is about 41 % of the cost for full labels. One could also look at it in another way: for fixed cross validation and test set, the train set can be made 8,3× larger with the same labelled cost.

values in [%]		**Predicted**					
		BG	**L1**	**L2**	**L3**	**L4**	**L5**
Actual	**BG**	99.48	56.56	29.81	29.5	29.22	41.93
	L1	0.09	39.65	3.73	0.08	0	0
	L2	0.11	3.76	63.36	2.42	0	0
	L3	0.11	0	3.1	63.49	2.91	0
	L4	0.11	0	0	4.5	65.91	1.85
	L5	0.11	0.03	0	0	1.96	56.22

Table 5.3: Confusion matrix for the model trained with pseudo label masks (network VGG16-FCN8 and cross-correlation loss), evaluated on the test set. The values have been normalized by the total number of voxels predicted in each class.

Since the final model is exactly the same network as the reference model, only trianed with pseudo masks instead of full masks, the inference time is exactly the same. Preprocesing of one volume was observed to take 3.9 seconds on average (see appendix B.2 for details on the hardware used). After this, evaluation of one volume and recombination of the crops takes, on average, 8,3 seconds. The segmentation masks illustrated in figure 5.4 can both be generated in 12,2 seconds.

The obtained segmentation masks are too restricted. While the position of the vertebrae seems to be predicted correctly, the model does not predict the full extend of the vertebra and many vertebra voxels are classified as background, hence the low recall. The ground cause of this phenomenon could not be discovered. No post-processing steps (dilation, watershed) to remedy this a-postiori were intended.

To investigate the performance difference observed between the different datasets, models were trained using only data from a single data source. Due to small differences between the datasets, both in labelling (some dataset include the laminae in the labels while others do not) and the imaging techniques used to obtain the dataset (CT or MRI), one could believe that combining these datasets could cause one to influence the network performance on the other. After training these single source models, large differences between the performances of these single source models were observed. The

latter calls for concluding that there are factors within each dataset determining how well the model can generalize.

Table 5.4: Summary table.

class-weighted dice score		Trained and evaluated on				
		all datasets	MyoSegmenTUM only	USiegen only	xVertSeg only	PLoS only
Reference		0.76				
Single- dim. (anotated on sagittal slices)	Transv.	0.53	0.64	0.71	0.62	0.70
	Frontal	0.41	0.46	0.37	0.41	
	Sagittal	0.34	0.36	0.28	0.36	
Pseudo mask model		0.55	0.58	0.39	0.52	

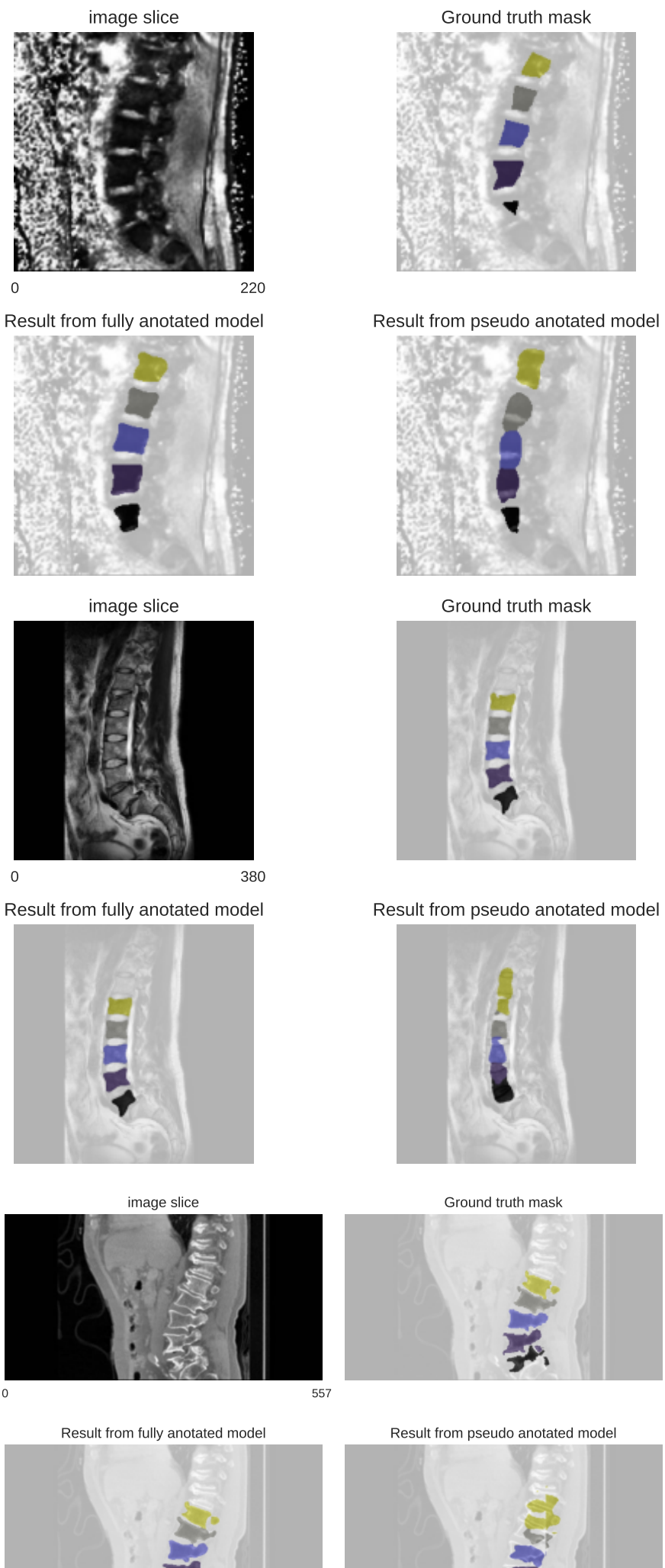


Figure 5.4: These images illustrate the difference in performance for the two models generated. On the top row, an image slice is shown next to the ground truth annotation mask. Two models were trained: the reference model, shown left on the second row and a model trained on pseudo masks of which the result is shown on the right side. These images show the final result of the procedure described in this thesis. The shown masks are not the pseudo masks themselves, they are the results obtained from the final model trained on these pseudo masks. From top to bottom, the image shows these results for slice 23 of MyoSegmenTUM volume 24, slice 24 of USiegen volume 12 and slice 260 of xVertSeg volume 14.
Colour legend:
● L1 ● L2 ● L3 ● L4 ● L5

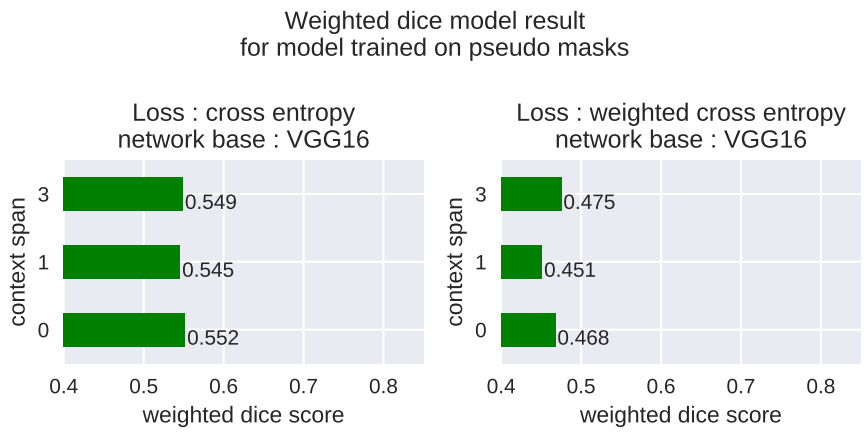
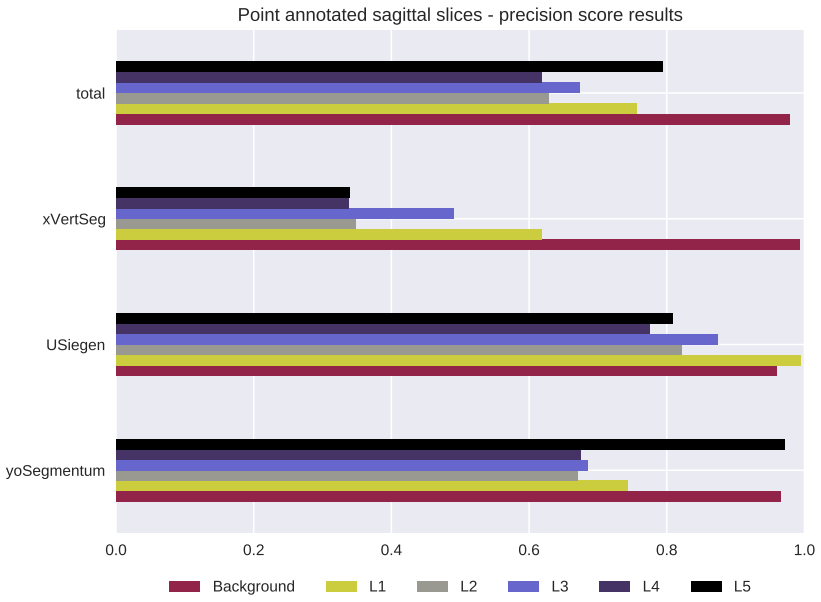


Figure 5.5: Class-weighted dice score for models trained on pseudo-masks.



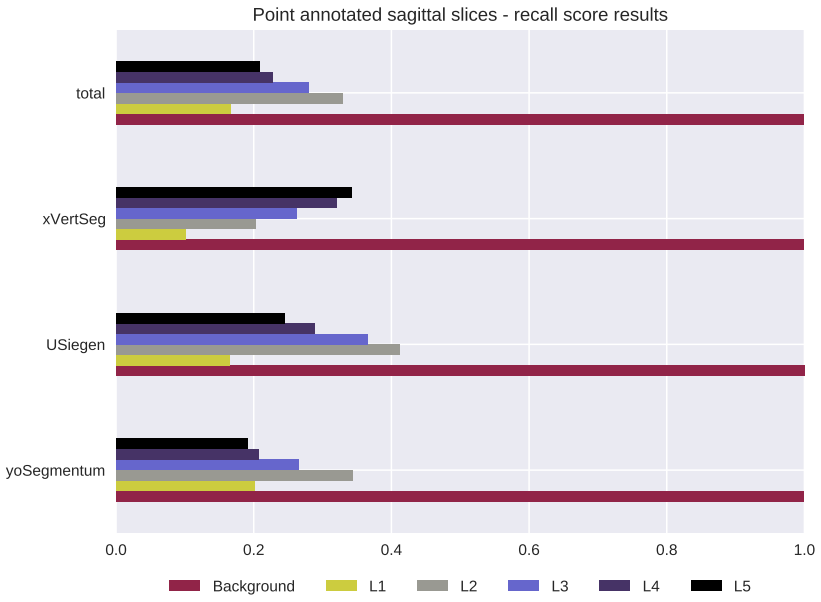


Figure 5.6: Precision and recall for the final model. From this image and from the illustrations shown in figure 5.4, it is clear that the obtained model does not seem to be able to predict the full extend of the vertebrae. The inferred masks are too restricted in dimensions.

Part IV

Conclusions

Abstract

This part of the document summarizes the most important conclusions from this work and possible ideas as a way forward after this work. The part starts with an evaluation and discussion of the labelling cost necessary to make the final model. Finally, the lessons learned from this project are laid out and some ideas for further development of weakly supervised models for medical applications are presented.

— 1 —

Conclusions

This research aimed to identify techniques for multi-label segmentation of volumetric scans of the human lumbar vertebra. The research was based on 5 publically available datasets, containing both CT and MRI scans. The solution proposed in this document is based on previous research, the consistency loss introduced in[22], extended with two new loss function components.

Further, the resulting models, trained with this technique are combined with a new algorithm to obtain pseudo masks with which the final model can be trained. This combination step was proven to be a valuable step to increase the segmentation mask performance, measured with the weighted dice metric.

The result of this work indicated that the developed techniques allow to obtain an inverse weighted dice score of 0.55 for a model trained with labels that represent 12 % of the cost of full label masks. This result should be compared to the inverse weighted dice score of 0.76 for a model that was trained on full label masks.

The results in this thesis have demonstrated that well-chosen additional loss terms can improve the performance of the consistency loss model and that the combination of the results of two-dimensional models trained on slices along a different dimensional axis can produce a three-dimensional segmentation mask of higher quality than the original masks.

— 2 —

Future work

The research presented in this work raises a number of unanswered questions and could inspire others to investigate this problem further. Personally, I believe the following subjects desirve further investigation and exploration:

Finer prior extrend: The prior extend loss introduced in this work[1] is based on a single maximal radius $r = 110mm$ of the lumbar vertebrae. This could be refined by imposing a different r value per vertebra or even by imposing different values along different axis of the vertebrae.

1. The prior extent loss is introduced on page 39

dataset difference: This work makes use of a collection of different publically available datasets. The difference in performance between these is striking. The developed model performance is unequal for the different datasets. This means that one cannot claim this model is generally applicable. It is only suitable for a limited range of scan types. It would be interesting to investigate how the model could be trained that is more generally applicable. Are there fundamental differences between the datasets that can be adressed with improved pre-processing? Can an adapted sampling scheme (under- or oversampling) help?

Weights for the point loss: This work demonstrated that a weighted point loss function does not improve the model performance. One could argue that the chosen weighing strategy was not optimal. Could it be a better idea to choose weights inverse proportional to the number of annotation points per class? Can another weighing scheme be conceived?

Optimal number of annotation points: This research did not conclude on determining the optimal number of class instance annotation points. There are two contradicting observations. First, there is the observation that the performance of the single dimension models decreases with more annotation points per class instance. On the other hand, decreasing the annotation points decreases the number of training slices for the other two slice dimensions on which no expert annotation was provided.

Prior position loss: One aspect of prior knowledge that has not been used in this work is the known sequence of the vertebrae. In [24], the known anatomical sequence of vertebrae was used for vertebrae identification. Although this proved not to be necessary in this work, one can wonder if including this information, for example in a way similar to [24], could help to reduce the model training time.

Post processing: The results obtained with the model trained on pseudo masks showed to systematically underestimate the extend of the vertebrae. It could be possible to remedy this in a post-processing step.

Part V

Appendix

Abstract

Apendices to the document:

1. Detail of the train test split.
2. Details regarding the reproducability of this work.
3. Agreement documents for use of the xVertSeg dataset.
4. Addentum regarding the pre-defence and the seminars followed during this academic year.

— A —

Train test split detail

The datasets in this research are both stratified and grouped (due to USiegen containing multiple scans from the same patient). These were splitted in a train, validation and test set taking into account these characteristics. This was discussed in chapter 1.2 on page 23. The result of this operation is detailed in the following tables.

Table A.1: Detailed table of the grouped and stratified split of the different dataset sources, taking into account the grouped nature of the USiegen dataset. Part 1 of 2

	patient	scan_id	split
0	MyoSegmenTUM_001	MyoSegmenTUM_001	train
1	MyoSegmenTUM_002	MyoSegmenTUM_002	xval
2	MyoSegmenTUM_003	MyoSegmenTUM_003	train
3	MyoSegmenTUM_004	MyoSegmenTUM_004	train
4	MyoSegmenTUM_005	MyoSegmenTUM_005	test
5	MyoSegmenTUM_006	MyoSegmenTUM_006	train
6	MyoSegmenTUM_007	MyoSegmenTUM_007	train
7	MyoSegmenTUM_008	MyoSegmenTUM_008	test
8	MyoSegmenTUM_009	MyoSegmenTUM_009	train
9	MyoSegmenTUM_010	MyoSegmenTUM_010	xval
10	MyoSegmenTUM_011	MyoSegmenTUM_011	train
11	MyoSegmenTUM_012	MyoSegmenTUM_012	train
12	MyoSegmenTUM_013	MyoSegmenTUM_013	train
13	MyoSegmenTUM_014	MyoSegmenTUM_014	train
14	MyoSegmenTUM_015	MyoSegmenTUM_015	train
15	MyoSegmenTUM_016	MyoSegmenTUM_016	train
16	MyoSegmenTUM_017	MyoSegmenTUM_017	train
17	MyoSegmenTUM_018	MyoSegmenTUM_018	test
18	MyoSegmenTUM_019	MyoSegmenTUM_019	train
19	MyoSegmenTUM_020	MyoSegmenTUM_020	train
20	MyoSegmenTUM_021	MyoSegmenTUM_021	xval
21	MyoSegmenTUM_022	MyoSegmenTUM_022	train
22	MyoSegmenTUM_023	MyoSegmenTUM_023	train
23	MyoSegmenTUM_024	MyoSegmenTUM_024	xval
24	MyoSegmenTUM_025	MyoSegmenTUM_025	train
25	MyoSegmenTUM_026	MyoSegmenTUM_026	xval
26	MyoSegmenTUM_027	MyoSegmenTUM_027	train
27	MyoSegmenTUM_028	MyoSegmenTUM_028	train
28	MyoSegmenTUM_029	MyoSegmenTUM_029	xval
29	MyoSegmenTUM_030	MyoSegmenTUM_030	test
30	MyoSegmenTUM_031	MyoSegmenTUM_031	test
31	MyoSegmenTUM_032	MyoSegmenTUM_032	xval
32	MyoSegmenTUM_034	MyoSegmenTUM_034	train
33	MyoSegmenTUM_035	MyoSegmenTUM_035	train
34	MyoSegmenTUM_036	MyoSegmenTUM_036	train
35	MyoSegmenTUM_037	MyoSegmenTUM_037	test
36	MyoSegmenTUM_038	MyoSegmenTUM_038	train
37	MyoSegmenTUM_039	MyoSegmenTUM_039	test
38	MyoSegmenTUM_040	MyoSegmenTUM_040	train
39	MyoSegmenTUM_041	MyoSegmenTUM_041	train
40	MyoSegmenTUM_042	MyoSegmenTUM_042	train
41	MyoSegmenTUM_043	MyoSegmenTUM_043	train
41	MyoSegmenTUM_043	MyoSegmenTUM_043	xval
42	MyoSegmenTUM_044	MyoSegmenTUM_044	train
43	MyoSegmenTUM_045	MyoSegmenTUM_045	train
44	MyoSegmenTUM_046	MyoSegmenTUM_046	train
45	MyoSegmenTUM_047	MyoSegmenTUM_047	train
46	MyoSegmenTUM_048	MyoSegmenTUM_048	train
47	MyoSegmenTUM_049	MyoSegmenTUM_049	test
48	MyoSegmenTUM_050	MyoSegmenTUM_050	train
49	MyoSegmenTUM_051	MyoSegmenTUM_051	test
50	MyoSegmenTUM_052	MyoSegmenTUM_052	xval

Table A.2: Detailed table of the grouped and stratified split of the different dataset sources, taking into account the grouped nature of the USiegen dataset. Part 2 of 2

	patient	scan_id	split
53	PLoS_003	PLoS_003	train
54	PLoS_004	PLoS_004	train
55	PLoS_005	PLoS_005	xval
56	PLoS_006	PLoS_006	test
57	PLoS_007	PLoS_007	train
58	PLoS_008	PLoS_008	train
59	PLoS_009	PLoS_009	train
60	PLoS_010	PLoS_010	train
61	PLoS_011	PLoS_011	xval
62	PLoS_012	PLoS_012	test
63	PLoS_013	PLoS_013	train
64	PLoS_014	PLoS_014	train
65	PLoS_015	PLoS_015	train
66	PLoS_016	PLoS_016	train
67	PLoS_017	PLoS_017	xval
68	PLoS_018	PLoS_018	test
69	PLoS_019	PLoS_019	train
70	PLoS_020	PLoS_020	train
71	PLoS_021	PLoS_021	xval
72	PLoS_022	PLoS_022	test
73	USiegen_001	USiegen_003	train
74	USiegen_001	USiegen_006	train
75	USiegen_001	USiegen_016	train
76	USiegen_002	USiegen_008	train
77	USiegen_002	USiegen_014	train
78	USiegen_002	USiegen_001	train
79	USiegen_002	USiegen_010	train
80	USiegen_002	USiegen_000	train
81	USiegen_004	USiegen_002	train
82	USiegen_007	USiegen_005	train
83	USiegen_007	USiegen_009	train
84	USiegen_008	USiegen_012	test
85	USiegen_009	USiegen_015	train
86	USiegen_010	USiegen_004	xval
87	USiegen_015	USiegen_007	train
88	USiegen_016	USiegen_013	train
89	USiegen_018	USiegen_011	test
90	xVertSeg_001	xVertSeg_001	train
91	xVertSeg_002	xVertSeg_002	train
92	xVertSeg_003	xVertSeg_003	train
93	xVertSeg_004	xVertSeg_004	train
94	xVertSeg_005	xVertSeg_005	train
95	xVertSeg_006	xVertSeg_006	train
96	xVertSeg_007	xVertSeg_007	xval
97	xVertSeg_008	xVertSeg_008	train
98	xVertSeg_009	xVertSeg_009	xval
99	xVertSeg_010	xVertSeg_010	train
100	xVertSeg_011	xVertSeg_011	test
101	xVertSeg_012	xVertSeg_012	train
102	xVertSeg_013	xVertSeg_013	train
103	xVertSeg_014	xVertSeg_014	test
104	xVertSeg_015	xVertSeg_015	xval

— B —

Used software & Reproducability of this research

B.1 Reproducability of the used environment

The reproducability of this work has been ensured in several ways:

Data : All datasets used in this project are publically available. For some datasets, it is required to request permission, like the author of this work has done.

Software : This project was performed completely in python, a publically available programming language. All packages used are open-source. To reproduce this research, no software has to be purchased.

Code : Both code an text of this project can be found on a public `GitHub` repository: `https://github.com/JanAlexanderPersonal/MastatThesis.git`. [1].

1. It is possible that small adaptations of the code are required to run it on other hardware. For example, to run it using a GPU with less internal memory it could be necessary to reduce the batch size.

Docker : The specific environment to run the code mentioned above can also be found in the mentioned GitHub repository in the form a dockerfile. It should be noted that this dockerfile requires a GPU that supports CUDA.

The above mentioned aspects should allow other researchers to reproduce the results in this document. This work makes use of parts of *haven.ai* [2] as a basis to manage different experiments. These experiments are conducted using the deep learning library *PyTorch*. This tool is widely used in boty industry and academic research. A list of other Machine Learning tools used in this work is provided below. Table B.1 is not an exhaustive list of all python libraries used in this work. It aims to list the more advanced tools, crucial for this work.

2. This is a collection of libraries developed by the team of dr. I. Laradji to manage different experiments in deep learning research.

Table B.1 is not an exhaustive list of all libraries used in this work. One can quickly dublicate the environment used in this work by building the *Docker* environment defined in `dockerfiles` (part of the `GitHub` repository).

B.2 Hardware

The main hardware components of the device used for this work are listed below:

Intel® Core™ i7-9700 CPU @ 3.00GHz × 8
RAM : 32 GB
GPU:
GeForce RTX 2080 Ti (4352 CUDA cores)
total memory 11264 MB
CUDA version 11.2

Library	version	reference
Python	3.8	Programming language
Deep learning and machine learning tools		
PyTorch	1.7.1	Deep learning library
scikit-learn	0.24.2	Machine learning toolbox
Tools to work with (medical) images		
SimpleITK	2.0.2	Library for medical images
kornia	0.2.0	Computer vision library linked to torch
scikit-image	0.18.1	Image manipulation library
Pillow	8.2.0	Image manipulation
Tools for matrix & scientific computation		
numpy	1.20.3	matrix manipulation
scipy	1.6.3	scientific calculation
Result visualization		
matplotlib	3.4.2	data visualization
seaborn	0.11.1	data visualization
Neural network experiment management		
haven		experiment management

Table B.1: Python libraries used.

B.3 Other

The formatting of this document, bringing Tufte layout elements in the LaTeX memoir class, is based on the excelent post found on `https://tex.stackexchange.com/questions/275565/tufte-layout-in-painless-memoir`. Didier Dufrasne gave a few very helpful tips for improving this layout. The Neural network architecture illustrations were made using this project: `https://github.com/HarisIqbal88/PlotNeuralNet`. The conceptual illustration of the neural network idea on page 7 is based on `https://tex.stackexchange.com/questions/153957/drawing-neural-network-with-tikz`.

— C —

Dataset agreements

Laboratory *of Imaging Technologies*
University *of Ljubljana*
Faculty *of Electrical Engineering*

To:

Name:	prof. dr. Tomaž Vrtovec
Address:	Laboratory of Imaging Technologies University of Ljubljana, Faculty of Electrical Engineering Tržaška cesta 25, SI-1000 Ljubljana, Slovenia
Phone:	+386 1 4768 783
E-mail:	tomaz.vrtovec@fe.uni-lj.si
Web:	http://lit.fe.uni-lj.si

From:

Name:	Jan Alexander
Address:	Sint-Lievenslaan 126 BE-9000 Gent Belgium
Phone:	0032478779156
E-mail:	jan.alexander@ugent.be

Subject:

Agreement on using the "Lumbar vertebra segmentation CT image database"

The Laboratory of Imaging Technologies (University of Ljubljana, Faculty of Electrical Engineering, Slovenia) will provide a password for accessing the "Lumbar vertebra segmentation CT image database". The database contains CT images of the lumbar spine for different subjects and the corresponding binary masks for each lumbar vertebra (i.e. L1, L2, L3, L4 and L5) obtained by manual segmentation. All images are in the near raw raster data (NRRD) file format. The database is stored on a password protected computer in the Laboratory of Imaging Technologies.

By signing this document I agree:

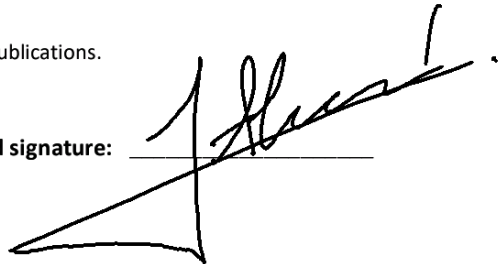
1) that I will not redistribute the provided password,
2) that I will not redistribute the database or its parts,
3) that I will include the following two references in any publication resulting from any use of this database:

   R1. Bulat Ibragimov, Boštjan Likar, Franjo Pernuš, Tomaž Vrtovec, "Shape representation for efficient landmark-based segmentation in 3D", IEEE Transactions on Medical Imaging, 33(4):861-874, 2014 [doi:10.1109/TMI.2013.2296976]

   R2. Robert Korez, Bulat Ibragimov, Boštjan Likar, Franjo Pernuš, Tomaž Vrtovec, "A framework for automated spine and vertebrae interpolation-based detection and model-based segmentation", IEEE Transactions on Medical Imaging, 34(8):1649-1662, 2015 [doi: 10.1109/TMI.2015.2389334]

   and will supply the data provider with copies of such publications.

Date: 22.11.2020 **Authorized signature:** _____



— D —

Predefence and seminars

On March 31st, 2021 the concept of this work was presented in a poster session seated by Prof. Stijn Vansteelandt. The poster used during this event is included on the next page for reference.

Addendum to Master thesis

Name student: **Jan Alexander**

Name promoter: **dr. Joris Roels**
dr. Bert Vankeirsbick

Thesis title:
Spine vertebrae segmentation in 3D CT scan images

Date **pre-defense**: March 31, 2021

Name and signature of one of the members of the Examination Board, testifying successful participation in the pre-defenses:

Prof. Stijn Vansteelandt

Date **seminar 1**: March 10, 2021

Title seminar 1:
Statistics and Epidemics: The role of statistics and modeling in understanding COVID-19 where we have been and the path forward, by Prof. dr. Nicholas Jewell

(Add details of the seminar on an extra page: speaker's name, title and abstract, time and location)
Signature of the promoter, testifying approval of and attendance of the seminar:

Date **seminar 2**: June 15, 2021

Title seminar 2:
BETS: The dangers of selection bias in early analyses of the coronavirus disease (COVID-19) Pandemic, by dr. Qingyuan Zhao

(Add details of the seminar on an extra page: speaker's name, title and abstract, time and location)
Signature of the promoter, testifying approval of and attendance of the seminar:

SPINE VERTEBRAE SEGMENTATION IN 3D CT SCAN IMAGES

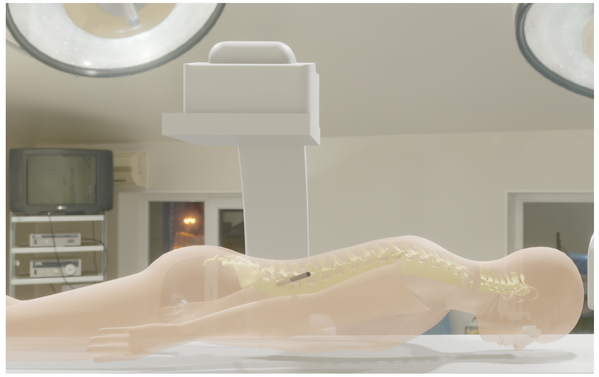
Jan Alexander
Promotors: dr. J. Roels & dr. ir. B. Vankeirsbilck

GHENT UNIVERSITY
FACULTY OF SCIENCES

PROBLEM MOTIVATION

CT scan segmentation

Automatization of CT scan interpretation:
pathology diagnosis and surgical intervention.



lumbar hernia procedure

Weak supervision

Data labelling costly and time consuming
What could we do with less?
Instance segmentation model trained on **point supervised** data can strongly reduce cost [1].

PROJECT OBJECTIVE

Construction and analysis of point supervised instance segmentation mode of the lumbar spine.
Comparison of the performance of point supervised models with fully supervised reference model.
Based on freely available data.

PREVIOUS WORK

Datasets

5 datasets [2-7,13]:

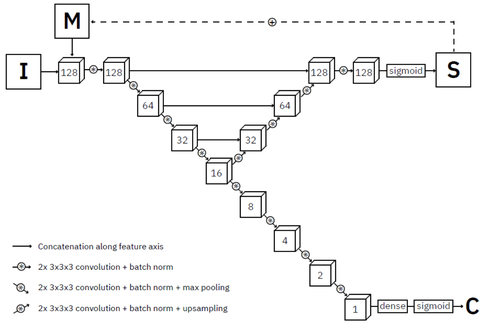
- 359 images, 242 patients
- 103 male, 109 female, 22 unknown
- Average age: 53 years

DATASET	IMAGES	ANNOTATION
USiegen	17	Full
xVertSeg	25	Full
PLoS	22	Full
MyoSeg	53	Full
Uwash	242	Point

Architectures

Fully supervised model as reference [12]

3D U-Net with instance memory



Point supervision techniques

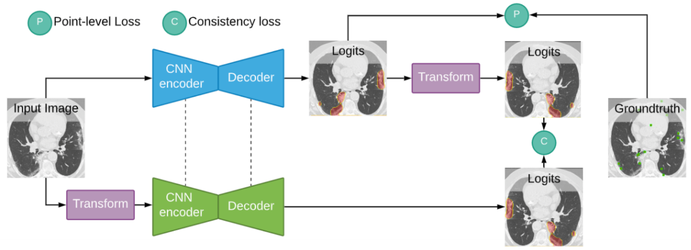


Illustration of consistency loss network [9]

APPROACH

Comparison with fully segmented model

Convert full segmentation to point supervision.

- Compare weakly supervised model with fully supervised model, trained on same dataset:

  Different backbones: ResNet, VGG16 & upsampling, U-net)

  Different loss functions: Consistency, WISe

  Evaluation of 3th dimension: 2D+ (channels), 3D, instance memory

- Train best model with UW data and evaluate again.

Aknowledgement

The original project idea occured to me due to the VLAIO – REISS project, a joint effort of Verhaert New Products and Services & Ugent IMEC.
The REISS project is led by dr. Ir. B. Braeckman

REFERENCES

[1]Bearman, A., Russakovsky, O., Ferrari, V., & Fei-Fei, L. (2015, jun). Whatś the Point: Semantic Segmentation with Point Supervision. Retrieved from http://arxiv.org/abs/1506.02106
[2]Burian, E., Rohrmeier, A., Schlaeger, S., Dieckmeyer, M., Diefenbach, M., Syväri, J., . . . Baum, T. (2019, 04). Lumbar muscle and vertebral bodies segmentation of chemical shift encoding-based water-fat MRI: The reference database MyoSegmenTUM spine. BMC Musculoskeletal Disorders, 20. doi:10.1186/s12891-019-2528-x
[3]Chu, C., Belavý, D. L., Armbrecht, G., Bansmann, M., Felsenberg, D., & Zheng, G. (2015, 11). Fully Automatic Localization and Segmentation of 3D Vertebral Bodies from CT/MR Images via a Learning-Based Method. PLOS ONE, 10, 1-22. doi:10.1371/journal.pone.0143327
[4]Glocker, B., Feulner, J., Criminisi, A., Haynor, D. R., & Konukoglu, E. (n.d.). Automatic Localization and Identification of Vertebrae in Arbitrary Field-of-View CT Scans. Tech. rep.
[5]Glocker, B., Zikic, D., Konukoglu, E., Haynor, D. R., & Criminisi, A. (n.d.). Vertebrae Localization in Pathological Spine CT via Dense Classification from Sparse Annotations. Tech. rep.
[6]Ibragimov, B., Likar, B., Pernuš, F., & Vrtovec, T. (2014, April). Shape Representation for Efficient Landmark-Based Segmentation in 3-D. IEEE Transactions on Medical Imaging, 33, 861-874. doi:10.1109/TMI.2013.2296976
[7]Korez, R., Ibragimov, B., Likar, B., Pernuš, F., & Vrtovec, T. (2015, Aug). A Framework for Automated Spine and Vertebrae Interpolation-Based Detection and Model-Based Segmentation. IEEE Transactions on Medical Imaging, 34, 1649-1662. doi:10.1109/TMI.2015.2389334
[8]Laradji, I. H., Rostamzadeh, N., Pinheiro, P. O., Vazquez, D., & Schmidt, M. (2019, jun). Instance Segmentation with Point Supervision. Retrieved from http://arxiv.org/abs/1906.06392
[9]Laradji, I. H., Rostamzadeh, N., Pinheiro, P. O., Vazquez, D., & Schmidt, M. (2020). Proposal-Based Instance Segmentation With Point Supervision. 2126-2130. doi:10.1109/icip40778.2020.9190782
[10]Laradji, I. H., Vazquez, D., & Schmidt, M. (2019, jul). Where are the Masks: Instance Segmentation with Image-level Supervision. Retrieved from http://arxiv.org/abs/1907.01430
[11]Laradji, I., Rodriguez, P., Mañas, O., Lensink, K., Law, M., Kurzman, L., . . . Nowrouzezahrai, D. (2020, jul). A Weakly Supervised Consistency-based Learning Method for COVID-19 Segmentation in CT Images. Retrieved from http://arxiv.org/abs/2007.02180
[12]Lessmann, N., van Ginneken, B., de Jong and Pim, A., & Išgum, I. (2018, apr). Iterative fully convolutional neural networks for automatic vertebra segmentation and identification. doi:10.1016/j.media.2019.02.005
[13]Zukić, D., Vlasák, A., Egger, J., Hořínek, D., Nimsky, C., & Kolb, A. (2014, 03). Robust Detection and Segmentation for Diagnosis of Vertebral Diseases Using Routine MR Images. Computer Graphics Forum, 33. doi:10.1111/cgf.12343

Bibliography

[1] Jiwoon Ahn, Sunghyun Cho, and Suha Kwak. "Weakly supervised learning of instance segmentation with inter-pixel relations". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2019-June (2019), pp. 2204–2213. ISSN: 10636919. DOI: 10.1109/CVPR.2019.00231. arXiv: 1904.05044.

[2] Muhammad M. Alam et al. "Lumbar morphometry: A study of lumbar vertebrae from a Pakistani population using computed tomography scans". In: *Asian Spine Journal* 8.4 (2014), pp. 421–426. ISSN: 19767846. DOI: 10.4184/asj.2014.8.4.421.

[3] Amy Bearman et al. "What's the point: Semantic segmentation with point supervision". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 9911 LNCS. Springer, Cham, June 2016, pp. 549–565. ISBN: 9783319464770. DOI: 10.1007/978-3-319-46478-7_34. arXiv: 1506.02106. URL: http://arxiv.org/abs/1506.02106.

[4] Hakan Bilen, Benenson Rodrigo, and Seong Joon oh. *ECCV 2020 Tutorial on Weakly-Supervised Learning in Computer Vision*. 2020. URL: https://github.com/hbilen/wsl-eccv20.github.io.

[5] Egon Burian et al. "Lumbar muscle and vertebral bodies segmentation of chemical shift encoding-based water-fat MRI: The reference database MyoSegmenTUM spine". In: *BMC Musculoskeletal Disorders* 20 (2019). DOI: 10.1186/s12891-019-2528-x.

[6] David Lacalle Castillo. *SemTorch · PyPI*. \url{https://pypi.org/project/SemTorch/}. Sept. 2020.

[7] Chengwen Chu et al. "Fully automatic localization and segmentation of 3D vertebral bodies from CT/MR images via a learning-based method". In: *PLoS ONE* 10.11 (Nov. 2015). Ed. by Dzung Pham, e0143327. ISSN: 19326203. DOI: 10.1371/journal.pone.0143327. URL: https://dx.plos.org/10.1371/journal.pone.0143327.

[8] Cheng-Hung Chuang et al. "Efficient Triple Output Network for Vertebral Segmentation and Identification". In: *IEEE Access* 7 (2019), pp. 117978–117985. ISSN: 2169-3536. DOI: 10.1109/access.2019.2934325.

[9] Özgün Çiçek et al. "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation". In: *CoRR* abs/1606.0 (2016). arXiv: 1606.06650. URL: http://arxiv.org/abs/1606.06650.

[10] Ben Glocker et al. "Automatic localization and identification of vertebrae in arbitrary field-of-view CT scans". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7512 LNCS (2012), pp. 590–598. ISSN: 16113349. DOI: 10.1007/978-3-642-33454-2_73.

[11] Ben Glocker et al. "Vertebrae localization in pathological spine CT via dense classification from sparse annotations". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 8150 LNCS. PART 2. 2013, pp. 262–270. ISBN: 9783642407628. DOI: 10.1007/978-3-642-40763-5_33.

[12] Ben Glocker et al. "Vertebrae localization in pathological spine CT via dense classification from sparse annotations". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8150 LNCS.PART 2 (2013), pp. 262–270. ISSN: 03029743. DOI: 10.1007/978-3-642-40763-5_33.

[13] B Ibragimov et al. "Shape Representation for Efficient Landmark-Based Segmentation in 3-D". In: *IEEE Transactions on Medical Imaging* 33.4 (Apr. 2014), pp. 861–874. ISSN: 1558-254X. DOI: 10.1109/TMI.2013.2296976.

[14] Rens Janssens and Guoyan Zheng. "Deep Learning based Segmentation of Lumbar Vertebrae from CT Images". In: 2 (2018), pp. 94–89. DOI: 10.29007/vt7v.

[15] Syed Ghufran Khalid et al. "Blood Pressure Estimation Using Photoplethysmography Only: Comparison between Different Machine Learning Approaches". In: *Journal of Healthcare Engineering* 2018 (2018). ISSN: 20402309. DOI: 10.1155/2018/1548647.

[16] Tobias Klinder et al. "Spine segmentation using articulated shape models". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 5241 LNCS. PART 1. 2008, pp. 227–234. ISBN: 354085987X. DOI: 10.1007/978-3-540-85988-8_28.

[17] R Korez et al. "A Framework for Automated Spine and Vertebrae Interpolation-Based Detection and Model-Based Segmentation". In: *IEEE Transactions on Medical Imaging* 34.8 (Aug. 2015), pp. 1649–1662. ISSN: 1558-254X. DOI: 10.1109/TMI.2015.2389334.

[18] Issam H. Laradji, David Vazquez, and Mark Schmidt. "Where are the masks: Instance segmentation with image-level supervision". In: *30th British Machine Vision Conference 2019, BMVC 2019*. July 2020. arXiv: 1907.01430. URL: http://arxiv.org/abs/1907.01430.

[19] Issam H. Laradji et al. "Instance Segmentation with Point Supervision". In: (June 2019). arXiv: 1906.06392. URL: http://arxiv.org/abs/1906.06392.

[20] Issam H. Laradji et al. "Proposal-Based Instance Segmentation With Point Supervision". In: *020 IEEE International Conference on Image Processing (ICIP)* 1 (2020), pp. 2126–2130. DOI: 10.1109/icip40778.2020.9190782. arXiv: arXiv:1906.06392v1.

[21] Issam H. Laradji et al. "Where Are the Blobs: Counting by Localization with Point Supervision". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11206 LNCS (2018), pp. 560–576. ISSN: 16113349. DOI: 10.1007/978-3-030-01216-8_34. arXiv: 1807.09856.

[22] Issam Laradji et al. "A Weakly Supervised Consistency-based Learning Method for COVID-19 Segmentation in CT Images". In: *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)* (Jan. 2021). arXiv: 2007.02180. URL: http://arxiv.org/abs/2007.02180.

[23] Issam Laradji et al. "A Weakly Supervised Region-Based Active Learning Method for COVID-19 Segmentation in CT Images". In: *arXiv* (July 2020). ISSN: 23318422. arXiv: 2007.07012. URL: http://arxiv.org/abs/2007.07012.

[24] Nikolas Lessmann et al. “Iterative fully convolutional neural networks for automatic vertebra segmentation and identification”. In: *Medical Image Analysis* 53 (Apr. 2019), pp. 142–155. ISSN: 13618423. DOI: 10.1016/j.media.2019.02.005. arXiv: 1804.04383. URL: http://arxiv.org/abs/1804.04383%20http://dx.doi.org/10.1016/j.media.2019.02.005.

[25] K. K. Maninis et al. “Deep Extreme Cut: From Extreme Points to Object Segmentation”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2018), pp. 616–625. ISSN: 10636919. DOI: 10.1109/CVPR.2018.00071. arXiv: 1711.09081.

[26] R. Austin McEver and B. S. Manjunath. “PCAMs: Weakly Supervised Semantic Segmentation Using Point Supervision”. In: (2020). arXiv: 2007.05615. URL: http://arxiv.org/abs/2007.05615.

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *CoRR* abs/1505.0 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[28] Anjany Sekuboyina et al. “A localisation-segmentation approach for multi-label annotation of lumbar vertebrae using deep nets”. In: *arXiv* 1 (2017), pp. 1–10. arXiv: 1703.04347.

[29] Ramprasaath R. Selvaraju et al. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. In: *International Journal of Computer Vision* 128.2 (2020), pp. 336–359. ISSN: 15731405. DOI: 10.1007/s11263-019-01228-7. arXiv: 1610.02391.

[30] Tong Shen et al. “Weakly supervised semantic segmentation based on web image cosegmentation”. In: *British Machine Vision Conference 2017, BMVC 2017*. 2017. ISBN: 190172560X. arXiv: 1705.09052.

[31] Eugenio Vocaturo, Ester Zumpano, and Pierangelo Veltri. “Image pre-processing in computer vision systems for melanoma detection”. In: *Proceedings - 2018 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2018* December (2019), pp. 2117–2124. DOI: 10.1109/BIBM.2018.8621507.

[32] Yunchao Wei et al. “Object region mining with adversarial erasing: A simple classification to semantic segmentation approach”. In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2017-January (2017), pp. 6488–6496. DOI: 10.1109/CVPR.2017.687. arXiv: 1703.08448.

[33] Ziv Yaniv et al. “SimpleITK Image-Analysis Notebooks: a Collaborative Environment for Education and Reproducible Research”. In: *Journal of Digital Imaging* 31.3 (2018), pp. 290–303. ISSN: 1618727X. DOI: 10.1007/s10278-017-0037-8.

[34] Dženan Zukić et al. “Robust Detection and Segmentation for Diagnosis of Vertebral Diseases Using Routine MR Images”. In: *Computer Graphics Forum* 33 (2014). DOI: 10.1111/cgf.12343.