



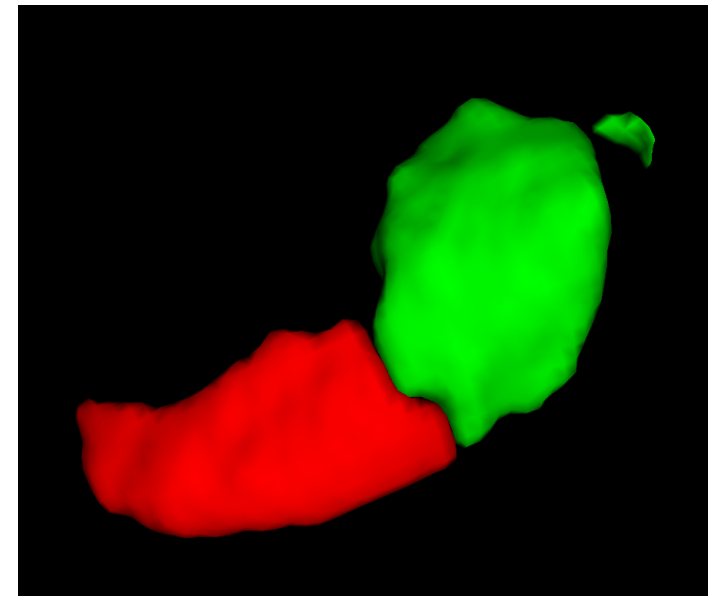
Jenny Stange Johansen, Mathias Aarseth Pedersen

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Computer Science

Jenny Stange Johansen
Mathias Aarseth Pedersen

Medical Image Segmentation: A General U-Net Architecture and Novel Capsule Network Approaches

June 2019





Norwegian University of
Science and Technology

Medical Image Segmentation: A General U-Net Architecture and Novel Capsule Network Approaches

Computer Science

Submission date: June 2019

Supervisor: Frank Lindseth

Norwegian University of Science and Technology
Department of Computer Science

Abstract

Medical imaging is an important tool for giving diagnoses and monitoring treatment response in health care. However, analyzing large amounts of image data is expensive because of the time and expertise required. In recent years, automatic image analyzing methods using artificial neural networks have made major advances. For image analysis, convolutional neural networks (CNN) have been particularly successful. Image segmentation methods aim to accurately indicate the location and shape of the structure in question. One commonly used architecture for medical image segmentation is a CNN called U-Net [54]. Capsule networks are an alternative to CNN that recently have shown promising results [56]. Further research has suggested improving capsule networks with matrix capsules and EM-routing, increasing its ability to capture part-whole relationships [26].

A U-Net variant was developed and applied to multiple medical segmentation tasks. The architecture, called 2.5D U-Net, proved to be versatile when applied to medical tasks, despite using no manual or automatic dataset-specific parameter tuning. Official test results obtained from Medical Segmentation Decathlon include Dice scores of 85% for spleen, 87% and 85% for the anterior and posterior hippocampus respectively, along with 90% and 54% for liver and liver tumour [60]. Results reported by LaLonde et al. using SegCaps on the LUNA16 lung segmentation were reproduced [38]. However, applying the architecture to other medical segmentation tasks gave disappointing results. It achieved Dice scores of about 50 and 30 percentage points less than 2.5D U-Net on the spleen and the heart datasets. The SegCaps architecture was also extended to perform multi-class segmentation; a task that has not previously been studied in the literature. The network was able to achieve at least as good scores as the original SegCaps when applied to the spleen and the heart datasets. Multi-SegCaps successfully segmented anterior and posterior hippocampus, achieving Dice scores of 72.4% and 70.5%. In the last experiment, SegCaps using EM-routing was implemented and applied to the hippocampus dataset. The use of EM-routing for image segmentation had not previously been demonstrated in the literature. The experiment was intended to be a proof of concept and was successfully able to perform binary segmentation of the two hippocampus classes, as a single target class, achieving a Dice score of 54.5%.

Sammendrag

I helsevesenet er medisinsk bildediagnostikk et viktig verktøy for å sette diagnoser og overvåke behandlingsrespons. Dessverre er analyse av de store mengdene bildedata kostbart på grunn av ekspertisen og tiden som kreves. I de siste årene har automatiske analyseringsverktøy ved bruk av kunstige nevralt nettverk gjort store fremskritt. For bildeanalyse har spesielt konvolusjonsnett (convolutional neural networks (CNN)) hatt stor suksess. Ved segmentering av bilder er utfordringen å presist definere posisjonen og formen på en predefinert struktur. En kjent arkitektur for medisinsk bildesegmentering er et CNN kalt U-Net [54]. Et nylig alternativ til CNN, kapselnettverk (capsule networks), har også vist lovende resultater [56]. Videre forskning har foreslått å forbedre kapselnettverk ved å benytte matrisekapsler (matrix capsules) og EM-ruting for å bedre nettverkets forståelse for helheten i bildet [26].

I dette prosjektet ble en U-Net variant utviklet og testet på flere medisinske datasett for segmentering. Arkitekturen, kalt 2.5D U-Net, viste seg å være robust for denne typen oppgaver, til tross for at den hverken brukte manuelle eller automatiske parameterjusteringer basert på typen datasett. Offisielle testresultater fra *Medical Segmentation Decathlon* inkluderer Dice scorer på 85% for milten, 87% og 85% for fremre and bakre hippocampus henholdsvis, i tillegg til 90% og 54% for lever og leversvulst [60]. Det neste eksperimentet reproduiserte resultatene som var rapportert av LaLonde et al. ved å benytte SegCaps for LUNA16 lunge-segmentering [38]. Derimot ga ikke arkitekturen like gode resultater på andre medisinske datasett. Den fikk Dice scorer rundt 50 og 30 prosentpoeng lavere enn 2.5D U-Net modellen på milten og hjertet, henholdsvis. Videre ble SegCaps utvidet til å håndtere flere klasser, det løste en utfordring som ikke tidligere var kjent løst i litteraturen. Nettverket oppnådde minst like gode Dice scorer som den originale SegCaps, da den ble brukt for segmentering av milten og hjertet. Multi-SegCaps viste evne til å segmentere flere klasser ved å oppnå Dice score på 72.4% and 70.5% for fremre og bakre hippocampus. I det siste eksperimentet ble SegCaps med EM-ruting implementert og brukt til segmentering av hippocampus. Forfatterne hadde ikke observert at EM-ruting for SegCaps var blitt brukt for segmentering i noe tidligere forskning. Eksperimentet var ment som en demonstrasjon av prinsippet og modellen viste suksessfullt evne til å gjøre binær segmentering av de to hippocampus klassene med en Dice score på 54.5%.

Preface

This thesis was written by two students during the spring semester of 2019, for the Department of Computer Science (IDI) at the Norwegian University of Science and Technology (NTNU).

We would like to thank our supervisor Frank Lindseth for allowing us to work on the exciting task of medical image segmentation. His expertise and suggestions during the project have been a key factor for success.

Trondheim, June 2, 2019

Jenny Stange Johansen and Mathias Aarseth Pedersen

Table of Contents

Abstract	i
Summary	ii
Preface	iii
Table of Contents	iv
List of Tables	viii
List of Figures	x
1 Introduction	1
1.1 Motivation	1
1.2 Project Description	1
1.3 Project Goals and Research Questions	2
1.4 Contributions	2
1.5 Report Structure	3
2 Background	4
2.1 Segmentation of Medical Imaging Data	4
2.1.1 Body Structures and Disease	4
2.1.2 Medical Imaging	10
2.1.3 Medical Image Segmentation	12
2.1.4 Medical Segmentation Challenges	13
2.2 Tools and Frameworks	15
2.2.1 Numpy	15
2.2.2 TensorFlow	15
2.2.3 Keras	16
2.2.4 Jupyter Notebook	16
2.2.5 ITK-SNAP	16
2.3 Computer Vision	17

2.3.1	Feature Extraction	18
2.3.2	Image Classification	18
2.3.3	Object Detection	18
2.3.4	Image Segmentation	18
2.4	Artificial Neural Networks	19
2.4.1	Weight Update	20
2.4.2	Activation Functions	20
2.4.3	Loss Functions	21
2.4.4	Learning Rate	22
2.4.5	Adam Optimizer	22
2.5	Generalization and Regularization	23
2.5.1	Early Stopping	23
2.5.2	Dropout	24
2.5.3	Weight Decay	24
2.5.4	Input Reconstruction	24
2.5.5	Data Augmentation	24
2.6	Convolutional Neural Networks	26
2.6.1	AlexNet	27
2.6.2	GoogLeNet	27
2.6.3	VGGNet	28
2.6.4	Residual Neural Networks	28
2.7	Image Segmentation Architectures	28
2.7.1	Fully Convolutional Networks	28
2.7.2	SegNet	29
2.7.3	U-Net	29
2.7.4	Mask R-CNN	30
2.8	Capsule Networks	30
2.8.1	Dynamic Routing	32
2.8.2	Matrix Capsules With EM-Routing	35
2.8.3	Segmentation	39
2.9	Previous Work	40
3	Methodology	45
3.1	Architectures	45
3.1.1	2.5D U-Net	45
3.1.2	SegCaps	47
3.1.3	Multi-SegCaps	49
3.1.4	EM-SegCaps	51
3.2	Datasets	53
3.2.1	LUNA16	54
3.2.2	Spleen	54

3.2.3	Heart	55
3.2.4	Hippocampus	55
3.2.5	Liver	56
3.2.6	Pancreas	57
3.2.7	BraTS	57
3.3	Preprocessing	58
3.3.1	Normalization	58
3.3.2	Augmentation	59
3.4	Experiments	62
3.4.1	Experiment 1: 2.5D U-Net for Medical Segmentation Tasks	62
3.4.2	Experiment 2: Reproduce Results from <i>Capsules for Object Segmentation</i>	62
3.4.3	Experiment 3: Multi-class Segmentation using SegCaps .	63
3.4.4	Experiment 4: SegCaps with EM-routing	64
3.5	Training	64
3.5.1	2.5D U-Net	65
3.5.2	SegCaps	65
3.5.3	Multi-SegCaps	65
3.5.4	EM-SegCaps	66
3.6	Evaluation	66
3.6.1	Metrics	66
3.6.2	Method	67
3.7	Environment	68
4	Results	69
4.1	2.5D U-Net for Medical Segmentation Tasks	70
4.1.1	Spleen	70
4.1.2	Heart	72
4.1.3	Hippocampus	73
4.1.4	Liver	75
4.1.5	Pancreas	77
4.1.6	BraTS	78
4.2	Reproduce Results from <i>Capsules for Object Segmentation</i>	80
4.2.1	LUNA16	80
4.2.2	Spleen	83
4.2.3	Heart	84
4.3	Multi-class Segmentation using SegCaps	85
4.3.1	Spleen	86
4.3.2	Heart	88
4.3.3	Hippocampus	89
4.3.4	Liver	91

4.3.5	BraTS	93
4.4	SegCaps with EM-routing	94
4.4.1	Binary Hippocampus	94
4.4.2	Multi-class hippocampus	96
5	Discussion	99
5.1	2.5D U-Net for Medical Segmentation Tasks	99
5.2	Reproduce Results from <i>Capsules for Object Segmentation</i>	102
5.3	Multi-class Segmentation using SegCaps	105
5.4	SegCaps with EM-routing	108
5.5	Reflections	111
6	Conclusion and Future Work	112
6.1	Conclusion	112
6.2	Future Work	112
	Bibliography	114
	Appendix	122
A	Additional Results	122
A.1	2.5D U-Net	122
A.2	SegCaps	123
A.3	Multi-SegCaps	124
B	NAIS Submission	127

List of Tables

2.1	The tasks of Medical Segmentation Decathlon	15
2.2	Commonly used activation functions.	21
2.3	Results from nnU-Net for phase 1	41
2.4	Results from nnU-Net for phase 2	41
2.5	A performance evaluation of SegCaps	43
3.1	The datasets used in the experiments	53
3.2	The input shapes used for hippocampus	56
3.3	The values used for normalization of the datasets	59
3.4	The augmentations used in the experiments	59
4.1	An overview of Dice scores for all models, part 1	69
4.2	An overview of Dice scores for all models, part 2	70
4.3	Dice scores from the test sets of MSD, using 2.5D U-Net	70
4.4	Scores for the spleen dataset using 2.5D U-Net	71
4.5	Split scores for the spleen dataset using 2.5D U-Net	71
4.6	Scores for the heart dataset using 2.5D U-Net	73
4.7	Split scores for the heart dataset using 2.5D U-Net	73
4.8	Scores for the hippocampus dataset using 2.5D U-Net	74
4.9	Split scores for the hippocampus dataset using 2.5D U-Net	74
4.10	Scores for the liver dataset using 2.5D U-Net	75
4.11	Split scores for the liver dataset using 2.5D U-Net	76
4.12	Scores for the pancreas dataset using 2.5D U-Net	77
4.13	Split scores for the pancreas dataset using 2.5D U-Net	78
4.14	Scores for the BraTS dataset using 2.5D U-Net	79
4.15	Split scores for the BraTS dataset using 2.5D U-Net	79
4.16	Scores for the LUNA16 dataset using SegCaps	81
4.17	Split scores for the LUNA16 dataset using SegCaps	81
4.18	Scores for the spleen dataset using SegCaps	83
4.19	Split scores for the spleen dataset using SegCaps	83
4.20	Scores for the heart dataset using SegCaps	84

4.21	Split scores for the heart dataset using SegCaps	85
4.22	Scores for the spleen dataset using Multi-SegCaps with Dice loss .	86
4.23	Scores for the spleen dataset using Multi-SegCaps with WCE-loss	87
4.24	Split scores for the spleen dataset using Multi-SegCaps	87
4.25	Scores for the heart dataset using Multi-SegCaps	88
4.26	Split scores for the heart dataset using Multi-SegCaps	88
4.27	Scores for the hippocampus dataset using Multi-SegCaps	89
4.28	Split scores for the hippocampus dataset using Multi-SegCaps . .	90
4.29	Scores for the liver dataset using Multi-SegCaps	91
4.30	Split scores for the liver dataset using Multi-SegCaps	91
4.31	Scores for the BraTS dataset using Multi-SegCaps	93
4.32	Split scores for the BraTS dataset using Multi-SegCaps	93
4.33	Scores for binary hippocampus segmentation using EM-SegCaps .	95
4.34	Split scores for binary hippocampus segmentation using EM-SegCaps	95
4.35	Scores for multi-class hippocampus segmentation using EM-SegCaps	96
4.36	Split scores for multi-class hippocampus segmentation using EM- SegCaps	96
A.1	Training time for the SegCaps models	123
A.2	The class weights used by WCE-loss for SegCaps	123
A.3	Split scores for the LUNA16 dataset before and after a bug was fixed	123
A.4	Split scores for the spleen dataset using Multi-SegCaps with Dice loss	124
A.5	The class weights used by WCE-loss for Multi-SegCaps on spleen	126

List of Figures

2.1	An illustration of the human brain	6
2.2	An illustration of human organs	8
2.3	Segmentation tasks in the BraTS 2018 challenge	14
2.4	A medical image visualized in ITK-SNAP	17
2.5	An illustration of common computer vision tasks	19
2.6	Elastic deformation on MNIST dataset	25
2.7	Salt and pepper noise applied to an image	26
2.8	An example of a convolutional neural network	27
2.9	A residual block	28
2.10	The original U-Net architecture	30
2.11	An illustration of a case where a CNN often would fail	31
2.12	A capsule detecting that its parts are in disagreement.	32
2.13	A capsule having several child capsules in agreement	33
2.14	The effect of changing latent variables of a capsule, illustrated on MNIST data	34
2.15	EM clustering of capsule predictions	36
3.1	A diagram of the 2.5D U-Net architecture	48
3.2	A diagram of the SegCaps architecture	50
3.3	A diagram of the EM-SegCaps architecture	53
3.4	A visualization of a volume from the LUNA16 dataset	54
3.5	A visualization of a volume from the spleen dataset	54
3.6	A visualization of a volume from the heart dataset	55
3.7	A visualization of a volume from the hippocampus dataset	56
3.8	A visualization of a volume from the liver dataset	57
3.9	A visualization of a volume from the pancreas dataset	57
3.10	A visualization of a volume from the BraTS dataset	58
3.11	An illustration of elastic deformation	60
3.12	Augmentations used in the experiments	61
4.1	Slices from segmentation of the spleen using 2.5D U-Net	72

4.2	A 3D visualization of spleen segmentation using 2.5D U-Net . . .	72
4.3	Slices from segmentation of the heart using 2.5D U-Net	73
4.4	A 3D visualization of hippocampus segmentation using 2.5D U-Net	74
4.5	Slices from segmentation of the hippocampus using 2.5D U-Net .	75
4.6	Slices from segmentation of the liver using 2.5D U-Net	76
4.7	Scores per volume for the liver dataset using 2.5D U-Net	77
4.8	Slices from segmentation of the pancreas using 2.5D U-Net	78
4.9	Slices from segmentation of brain tumour using 2.5D U-Net	79
4.10	Slices from segmentation of the lungs using SegCaps	81
4.11	Slices from segmentation of the lungs where the label is incorrect	82
4.12	Scores per volume for the LUNA16 dataset using SegCaps	82
4.13	Slices from segmentation of the spleen using SegCaps	84
4.14	Slices from segmentation of the heart using SegCaps	85
4.15	Slices from segmentation of the spleen using Multi-SegCaps	87
4.16	Slices from segmentation of the heart using Multi-SegCaps	89
4.17	Slices from segmentation of the hippocampus using Multi-SegCaps	90
4.18	A 3D visualization of hippocampus segmentation using Multi-SegCaps	91
4.19	Slices from segmentation of the liver using Multi-SegCaps	92
4.20	Scores per volume for the liver dataset using Multi-SegCaps	92
4.21	Slices from segmentation of brain tumour using Multi-SegCaps . . .	94
4.22	Slices from binary segmentation of the hippocampus using EM-SegCaps	95
4.23	Slices from multi-class segmentation of the hippocampus using EM-SegCaps, for split 0	97
4.24	Slices from multi-class segmentation of the hippocampus using EM-SegCaps, for split 3	98
A.1	An activation map for a 2.5D U-Net model trained with Dice loss .	122
A.2	The loss graph for a Multi-SegCaps model with Dice loss	124
A.3	Slices from segmentation of the spleen using Multi-SegCaps with Dice loss	125
A.4	An activation map for a Multi-SegCaps model trained with Dice loss	126

Chapter 1

Introduction

This chapter gives an introduction to the project. It addresses the motivation behind the task and describes the project goals along with the research questions. At last, the contributions from this work are presented.

1.1 Motivation

Medical imaging is an important tool for giving patients diagnoses and monitoring treatment response. Analyzing medical images requires expert knowledge and is a time-consuming task. Automation of the process can save both time and costs; resources which are often needed elsewhere in health care. Medical image segmentation aims to precisely determine the location and shape of the body part or structure in question. With correct diagnosis and treatment, patients may live longer and have a higher quality of life.

In recent years, automatic image segmentation using deep learning has shown great results. The improvement due to convolutional neural networks is one of the main reasons for the recent progress in the field of image analysis. Even though medical image segmentation has made great strides, it is still a complex task. Continued research in this area is important, both in terms of improving state of the art methods and investigating methods that challenge the current image segmentation paradigm.

1.2 Project Description

A variant of the U-Net architecture, inspired by Isensee et al., will be implemented and trained for solving several medical segmentation tasks [30]. How general the architecture, called 2.5D U-Net, is with regard to learning medical segmentation

tasks will be investigated. The segmentation architecture SegCaps, based on capsule networks, will be applied to lung segmentation for reproducing the experiment by LaLonde et al. [38]. Further, SegCaps will be applied to several medical segmentation datasets and extended to handle the segmentation of multiple classes. The performance of using the SegCaps architectures will be compared to the performance of the U-Net variant. At last, a new type of network for segmentation based on capsule networks and the EM-routing algorithm will be developed [26]. The feasibility of matrix capsules for image segmentation will be addressed, both for binary as well as for multi-class segmentation.

1.3 Project Goals and Research Questions

The goal of this thesis will be to determine if 2.5D U-Net is suitable for medical segmentation tasks in general. Also, the applicability of SegCaps to similar tasks will be addressed. This includes reproducing the results of the original SegCaps paper and extending the architecture into performing segmentation of multiple classes. By implementing and executing experiments with the use of EM-routing for SegCaps, the goal is to predict if this method has a future within the field of segmentation. The following research questions (RQs) will be addressed:

- RQ1** How well does the 2.5D U-Net architecture perform on medical segmentation tasks in general?
- RQ2** Is the method and results in *Capsules for Object Segmentation* by LaLonde et al. reproducible? How well does SegCaps perform on similar datasets?
- RQ3** Can SegCaps be adapted to multi-class segmentation, giving results comparable to the original architecture? Is SegCaps able to compete with a U-Net based model?
- RQ4** Is image segmentation using SegCaps with EM-routing possible, and does it show promising results?

1.4 Contributions

A 2.5D architecture based on an U-Net variant was implemented and evaluated on multiple medical segmentation tasks. Segmentations of the test sets for six out of ten datasets from the Medical Segmentation Decathlon (MSD) were submitted to the competition. The obtained results were directly comparable to state of the art methods in this field of research. Despite not being able to beat the state of the art, the model showed potential and future research should address task-specific

automatic preprocessing and postprocessing steps to improve it further.

The method and results from *Capsules for Object Segmentation* were reproduced. The experiment showed that the architecture was, as reported in the paper, capable of segmenting images of lungs. However, the experiment also revealed that the architecture struggles to adapt to other medical datasets.

The SegCaps architecture was further developed to allow for multi-label image segmentation. It was experimentally shown to be able to perform segmentation of datasets with multiple label classes, but struggles to segment highly irregular structures in imbalanced datasets. An extension of SegCaps allowing for segmentation of multiple classes is an experiment not previously shown in the literature, as far as the authors know.

A framework for image segmentation using capsule networks and the EM-routing algorithm was developed and evaluated. As far as the authors are concerned, an attempt of this has never been reported in the literature so far. Although the model did not give very good results, it showed that it was capable of segmenting some simple structures from the hippocampus dataset. The experiment revealed that capsule networks with EM-routing can be expected to learn segmentation on some level, in the future, given that the algorithms underneath are optimized to be less demanding in terms of computational and memory requirements.

A position paper about this thesis and previous work by the authors was submitted to the NAIS 2019 Symposium¹ (Appendix B). The paper outlined some of the methods that were applied to a selection of segmentation tasks, along with the results achieved at the time. It was accepted as a poster for the conference.

1.5 Report Structure

This report consists of six chapters. The first chapter gives an introduction to the project and the motivation behind it. The required background knowledge, from which the project is built, is presented in the second chapter. The third chapter describes the experiments and the method used for answering the research questions. After the methodology is presented, the result chapter shows the outcome of the experiments. The fifth chapter discusses the results and reflects upon lessons learned during this project. The last chapter summarizes the discussion of the results in a conclusion and suggests future work.

¹<https://www.nora.ai/news-and-events/news/2019-nais-symposium.html>

Chapter 2

Background

This chapter introduces the theory behind the research conducted in this thesis. At first, it gives an introduction to the segmentation of medical images and presents the tools used. Further, computer vision and artificial neural networks (ANN) are addressed. The next sub-chapter presents regularization and generalization techniques commonly applied to ANNs. A type of ANN frequently used for image analysis, convolutional neural networks (CNN), is presented. Some common image segmentation architectures are looked into and a recently developed alternative to CNN, capsule networks, are addressed. At last, previous work presents specific research and results that are related to this thesis.

2.1 Segmentation of Medical Imaging Data

Segmentation of medical imaging data aims to accurately determine the location and shape of the body part or structure in question. The imaging data may contain different body structures of interest, and the goal of analyzing them is usually to detect abnormalities and disease, such as tumours. In this section, relevant body structures and common medical imaging tools are presented. Further, the task of medical image segmentation along with related segmentation challenges are introduced.

2.1.1 Body Structures and Disease

This sub-section presents body structures and common diseases related to each of them. The body structures addressed are those that are relevant to this project. These are the brain, a region of the brain called the hippocampus, the heart, the lungs, the liver, the spleen, and the pancreas. The body parts are visualized in

Figure 2.2 and Figure 2.1. The last part of the section looks at tumours in general, which may be found in all of the mentioned body parts.

Brain

The central nervous system consists of the brain and the spinal cord [51]. The main task of the central nervous system is to control activity in the other organs of the body. The brain is responsible for processing information received from sensory input, making decisions, and sending instructions to the rest of the body.

The brain can be divided into three parts: the cerebrum, the cerebellum, and the brain stem. The cerebrum is the largest part and consists of white and grey matter. Further, it is divided into two cerebral hemispheres, which are the left and the right parts of the brain. In humans, the cerebellum is located at the lower back of the head and is important for balance and translating perceptions into muscle movement. The brain is connected to the spinal cord by the brain stem. Thus, the brain stem is crucial for transferring signals in and out of the brain. It is also involved in the regulation of cardiac and respiratory function.

The brain is protected by the skull and cerebrospinal fluid, and is separated from the bloodstream. Despite being protected, the brain is subject to disease, damage, and infections. A stroke can be caused by bleeding or a lack of blood flow. Examples of brain disease causing progressive loss of functioning neurons are Parkinson's disease and types of dementia. Mental disorders such as depressions and schizophrenia are associated with brain dysfunctions. Abnormal growths that form in the brain or spinal cord are called brain tumours. Abnormalities in the brain are often investigated by MRI, CT and PET scans, along with EEG measurements.

A cancerous tumour that starts in the brain is called a primary tumour, while a brain tumour that is caused by spread from another part of the body is called a metastatic brain tumour. The most common type of primary brain tumour is glioma [20]. It counts for around half of the cases each year, and 80% of all malignant brain tumours. Of all brain tumours, it makes up approximately 30%. Gliomas are usually of WHO grade II-IV. Gliomas of higher grades almost always grow back after surgical excision. It is usually not possible to cure a glioma, and the average survival rate for high-grade gliomas is only a couple of years. The brain is responsible for many important body functions, and a brain tumour will cause signs and symptoms. These are highly dependent on the size and position of the tumour. Medical imaging is central to the diagnosis of brain tumours.

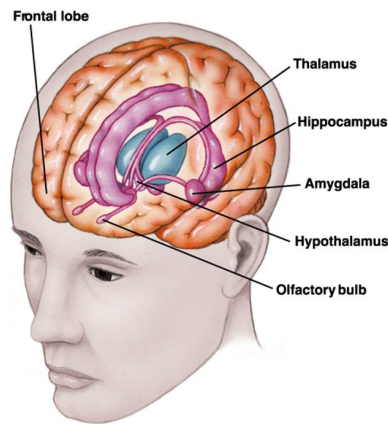


Figure 2.1: An illustration of the human brain.

Source: <https://aybu.edu.tr/sinancanan/contents/files/411limbic-hipot.pdf>

Hippocampus

The hippocampus is located under the cerebral cortex, which is the outer layer of grey matter in the brain [29]. Humans have two hippocampi, one on each side of the brain. It is important for processing of impressions from short-term memory to long-term memory. The hippocampus also plays a main role in orientation, which is controlled by spatial memory. The hippocampus is one of the few places in the brain where nerve cells continue to be created in adults.

The hippocampus is one of the first parts of the brain to suffer damage during Alzheimer's disease. Signs of disease are often impaired memory and spatial orientation. However, this can also indicate other forms of dementia. Further research has shown that hippocampus is likely to be related to schizophrenia and post-traumatic stress disorder (PTSD) [50]. Alzheimer, stress, and depression are plausible to be linked to a smaller sized hippocampus.

Heart

The human heart is a muscle located in the chest [3]. It is responsible for pumping blood through the circulatory system of the body. The blood transports oxygen and nutrients to organs and brings waste products away. The heart consists of four chambers, where both the left and the right part consist of upper atria and lower ventricles. The atria receive blood from the body and lungs pumping it to the ventricles which in its turn pumps it into the lungs and body. Pacemaking cells create a rhythm in which the heart pumps the blood.

Cardiovascular diseases are the leading cause of deaths worldwide [12]. The term includes coronary heart disease, heart attack, stroke, vascular disease, and peripheral arterial diseases. Risk factors are age, gender, smoking, diabetes, high blood pressure, obesity and lack of physical activity, among others. Cardiac imaging can address the shape and functioning of the heart. CT, MRI, PET, and ultrasound are frequently used for this purpose.

Lungs

In humans, the lungs are located in the chest on either side of the backbone and are the primary organ of the respiratory system [66]. The lungs are responsible for bringing oxygen from inhaled air into the blood and extracting carbon dioxide from the blood for releasing it. The right lung is bigger, as the left one shares space with the heart.

The lungs are subject for several diseases, such as infections, inflammations and cancer. Inflammation is often caused by infection due to bacteria or viruses. Pneumonia is the name of inflammation in the lung tissue. The respiratory tract and the pleurae surrounding the lungs can also be subject to inflammation. In other cases, an inflammation in the lung is called pneumonitis. Cancer can occur directly in a lung or spread from other body parts. In particular, smoking drastically increases the risk of lung cancer. Scans as CT is common for detection of cancer and other lung diseases.

Liver

The liver is a central organ for digestion and has many functions [66]. It produces proteins, bile and regulates lots of biochemical reactions. It also cleanses the blood by breaking down waste products, alcohol and old blood cells. The liver is located at the upper right part of the abdomen.

The liver is subject for disease, which can threaten the vital functions it is responsible for. Hepatitis is the name of inflammation of the liver and is most commonly caused by viruses. There exist several liver disorders which are caused by excessive alcohol consumption, grouped as alcoholic liver diseases. Liver cancer is most commonly developed after liver damage from hepatitis B, hepatitis C or alcohol [62].

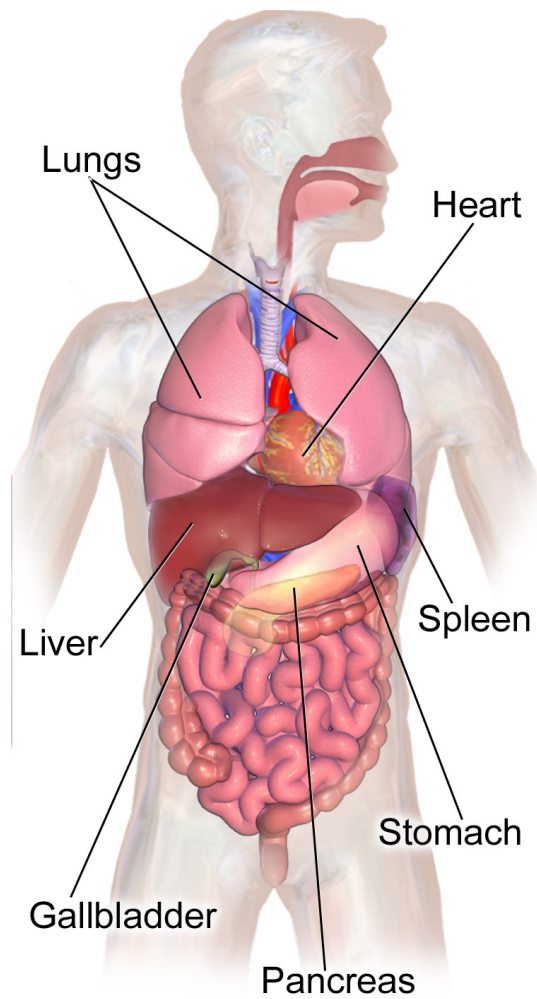


Figure 2.2: An illustration of human organs.

Source: https://commons.wikimedia.org/wiki/File:Abdominal_Organs_Anatomy.png

Spleen

The spleen is responsible for cleansing the blood from foreign substances, organisms, particles and old or damaged red blood cells [66]. It also recycles iron. These functions are primarily performed by white blood cells. In case of trauma to large bones, the spleen might start producing red blood cells, which is the same function as it had before birth.

Splenomegaly is a disease where the spleen is enlarged. It might be caused by several conditions including tumours and leukaemia. The absence of normal spleen function is called asplenia and can be caused by the lack of a spleen at birth, Sickle-cell disease or surgically removing the spleen.

Pancreas

The pancreas has two functions, it creates pancreatic juice in the digestion and it regulates blood sugar levels [66]. It is located behind the stomach, in the abdomen. The pancreatic juice neutralizes acid and breaks down carbohydrates, proteins, and fats in food entering the duodenum from the stomach. Cells in the pancreas produce glucagon, which increases blood glucose levels and helps regulate blood sugar.

The pancreas is subject to diseases as inflammation, cancer and diabetes. Pancreatitis is inflammation in the pancreas, which can be acute or chronic [68]. It is often caused by recurrent gallstones or chronic alcohol use. Pancreatic cancer is one of the top 5 causes of death from cancer [68]. Symptoms are often shown at a stage where it is too late for surgery, and only palliative treatment can be given. There are two types of diabetes where Diabetes mellitus type 1 is a disease where the immune system attacks the cells that create insulin. Diabetes mellitus type 2 is usually a combination of insulin resistance and reduced insulin production.

Tumours

A tumour is an uncontrolled, abnormal growth of cells or tissues in the body [13]. A tumour can be either malignant/cancerous or benign, where the main difference is that cancerous tumours might spread to other parts of the body. Benign tumours are usually not harmful and grow slowly. A third type is called precancerous, that are cells that might grow into cancer if they are not treated.

Cancer could give many different symptoms including a lump, abnormal bleeding, unexplained weight loss, prolonged cough and change in bowel movements [11].

Many types of cancer can be avoided by eating healthy, exercising, avoiding tobacco and excessive alcohol consumption, and taking the recommended vaccines. Despite this, genetics play an important role and some types of cancers are caused by inherited defects [18]. Survival is highly dependent on the type of cancer and when treatment is started.

Symptoms and signs, along with screening tests, are important for recognizing tumours [22]. Often medical imaging is used for investigating further. Blood samples are often used, and for some types of cancer substances called tumour markers can be found in body fluids. These might indicate development or recurrence of cancer. Finally, an examination of cells under a microscope by a pathologist is commonly used to confirm cancer. These cells are usually taken from the body by local anaesthesia.

A tumour can be classified by the WHO Grading System [44]. The system grades the tumour as one of four categories, where the highest grade is the most serious. Grade I tumours look like normal cells, spread slowly and rarely spread to nearby tissues, while Grade IV tumours usually are incurable and spread quickly.

2.1.2 Medical Imaging

Medical imaging is concerned with creating images of inside the human body, both to examine internal structures and its functioning. The goal is to identify abnormalities for diagnosing and treating disease, where radiology is the name for achieving this by using medical imaging. Medical imaging technologies often utilize aspects from biomedical engineering, medical physics, and computer science for creating images. Imaging tools that are commonly used include x-rays, computed tomography, magnetic resonance imaging, positron emission tomography and ultrasound.

Magnetic resonance imaging

One of the most common techniques for imaging is magnetic resonance imaging (MRI) [27]. MRI uses magnetic fields and frequencies in the radio wave spectrum to create images of body tissue. Atoms are made of protons, neutrons, and electrons, which all have a property known as spin. Without any external force, these spin vectors are randomly oriented and cancel each other out. When applying a strong magnetic field, the spin vectors for all the atoms align to the magnetic field, either in parallel, which is a low energy state or anti-parallel, its high energy state. A greater proportion of atoms are in their low energy state, giving a net magnetic vector in the direction of the magnetic field.

Hydrogen nuclei of tissue get excited away from its equilibrium when hit by radio-frequency radiation and start spinning in another direction. After a burst of radiation, the hydrogen's spin vector will relax back towards its equilibrium state while emitting photons. The time required for the molecules of a given tissue to return to their equilibrium is known as its T1 or longitudinal relaxation time. A majority of the nuclei will initially spin in-phase when hit with a burst of radiation, given that the radiation frequency matches the rotation frequency. Eventually, due to interactions between particles and magnetic field inhomogeneity, the particles will dephase. The T2, also known as transverse relaxation time, is the amount of time until transverse magnetization has fallen to approximately 37%.

The imaged tissue can be determined by examining the signal emitted at a certain time (echo time) and comparing it to the expected relaxation time of different tissues. By varying the intervals between radio-frequency pulses and the time the signal received are examined, it is possible to create images which exaggerate different properties of different tissue. For instance, by lowering the time between RF-pulses, the properties of the T1 relaxation are more visible, giving a T1-weighted image.

Images are made by capturing the electromagnetic radiation that is emitted by the tissue and later postprocessing them. Different modalities of MRI images that exaggerate different tissues are made by varying the frequency of RF-bursts sent to the tissue. Most common relaxation modalities are T1-weighted, T2-weighted and FLAIR. MRI imaging can be enhanced by using a contrast solution, e.g. gadolinium, which will change the relaxation properties of some tissues under certain conditions.

Computed tomography

Computed tomography scans (CT) are used to visualize the inside of objects [7]. CT most commonly refers to measurements achieved by using x-rays, but other methods exist such as positron emission tomography (PET) and single-photon emission computed tomography (SPECT). In CT, the measurements are acquired from different angles and processed by a computer into a three-dimensional volume. The images are sampled using rays that rotate around the object. Variations in images are caused by different absorption of x-rays in different tissues. Tissue density is measured in the unit Hounsfield (H).

On the Hounsfield scale, the air is represented by a value of 1000, water has the value 0 and compact bone has the value around 1000 [8]. By using this scale,

different organs are found at specific image ranges. By using an organ-specific level and window when inspecting the CT images, only the relevant information is shown. The window width selects the range of Hounsfield units for an image, and the window level determines the centre Hounsfield unit in this range.

CT images of the head are often used to detect tumours, infarction, haemorrhage and bone trauma. However, MRI is in several head disorders preferred over CT due to the higher information gain from MRI images and the ionizing radiation caused by CT. CT scans are also often used for acquiring lungs images, as regular x-ray images usually do not show chronic or acute changes to the lungs. For lungs, high-resolution CT is usually obtained by scanning both in inspiration and expiration.

Advantages of using CT for imaging include the elimination of structures outside the area of interest and the ability to distinguish the tissues with a low difference in density. However, the radiation from x-rays might damage cells in the body. The amount of radiation is determined by volume scanned, patient build, patient age, number and type of scan sequences, and desired resolution and image quality [70]. A high amount of radiation might lead to radiation-induced cancer, however, the increased risk from performing a single CT scan is estimated to be negligible [46].

2.1.3 Medical Image Segmentation

The goal of medical image segmentation is to accurately indicate the shape and location of predefined structures in an image. The structures can, for example, be tumours or body parts as the spleen, hippocampus, heart, colon, liver or pancreas. Because of the individual variability of tissues in size, shape, and appearance, medical segmentation is a difficult task.

Medical image segmentation can be categorized into manual, semi-automatic and automatic segmentation [21]. In manual segmentation, the full segmentation is performed by a human. This has been the standard for medical segmentation for a long time, because of the difficulty of the tasks. Manual segmentation is time-consuming and has to be performed by an expert, thus it is a costly process. Semi-automatic segmentation requires some degree of human interaction, which is supported by computer software. An example of a semi-automatic approach is when an expert needs to manually make a segmentation for a 2D slice as input, and the algorithm outputs the 3D segmentation. When the full segmentation is done by computer software, it is called automatic or fully-automatic segmentation.

Automated segmentation might reduce the time spent by experts for analyzing the images and help indicate abnormalities. Further, comparison of segmentation from different times could help to analyze treatment response or discover abnormal changes. Segmentation of body structures might support diagnostics or treatment. By segmenting high-resolution images experts can get an accurate indication of the boundaries of the tissue. An example application is for treatment of tumours where the segmentation is needed for radiation or surgery. Additionally, several classes can be segmented, such as different parts of a tumour. In general, systems that assist doctors in analyzing medical images are called computer-aided detection systems.

2.1.4 Medical Segmentation Challenges

Lung Nodule Analysis 2016

The Lung Nodule Analysis 2016 (LUNA16) dataset consists of 888 CT scans of lungs [4]. The challenge is motivated by the reduction of lung cancer by computer-aided detection of nodules. Lung nodules are spots in the lungs that are 3 centimetres in diameter or less, which could indicate lung cancer, inflammation or infection. The annotations exist of nodules that are at least 3 millimetres. Each annotation was approved by at least 3 radiologists.

There are two tasks in the LUNA16 challenge. The first task is nodule detection by assigning a probability of nodule to each location in the image. The second task concerns false positive reduction by assigning a probability of being a nodule to a predefined list of possible nodule locations. In addition to these tasks, the LUNA16 dataset contains computer-generated labels for segmentation of the left and right lung.

Multimodal Brain Tumour Image Segmentation Benchmark

The Multimodal Brain Tumour Image Segmentation Benchmark (BraTS) is a challenge of segmenting glioma brain tumours using MRI scans [48]. Since 2012 there has been a yearly evaluation of participants. Each year a dataset for training is made available and the test set is kept hidden until the final evaluation of the developed models.

The dataset consists of multimodal MRI scans, acquired from different locations. The scans are from glioma patients. The MRI modalities used are T1w, T1Gd, T2w, and FLAIR. The participating models are evaluated using the Dice score

[14], precision, also known as the positive predictive value, and recall, also known as sensitivity. The main task in BraTS 2018 asks the participants to do segmentation on sub-regions of glioma tumours, see Figure 2.3. The regions are the enhancing tumour, the tumour core, and the whole tumour.

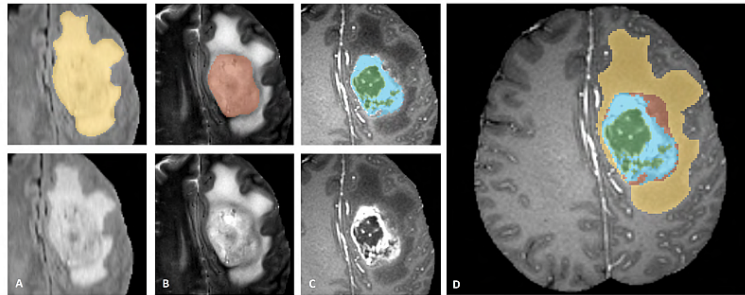


Figure 2.3: The sub-regions of the segmentation task in the BraTS 2018 challenge: the whole tumour, the tumour core and the enhancing tumour shown in yellow, red and blue/green respectively.

Source: The Multimodal Brain Tumour Image Segmentation Benchmark [48]

Medical Segmentation Decathlon

The Medical Segmentation Decathlon (MSD) is an image segmentation challenge aiming to produce general purpose image segmentation algorithms [60]. MSD consists of 10 datasets with various target structures. The goal is an algorithm performing well on different tasks without using human-defined task-specific model parameters. A list of the tasks can be seen in Table 2.1.

In the first part of the challenge, the participants are asked to develop a model suitable for learning 7 given segmentation tasks. The developed model is trained and tested on all these tasks. In the second part of the challenge, 3 new datasets are revealed, and the winners are the ones with the overall best performance on all 10 datasets.

Task	Modality	Target classes	Train images	Test images
Liver	CT	2	131	70
Brain	Multimodal MRI	3	484	266
Hippocampus	MRI	2	263	131
Lungs	CT	2	64	32
Prostate	Multimodal MRI	2	32	16
Heart	MRI	1	20	10
Pancreas	CT	2	282	139
Colon	CT	1	126	64
Hepatic Vessel	CT	2	303	140
Spleen	CT	1	41	20

Table 2.1: The tasks of Medical Segmentation Decathlon.

2.2 Tools and Frameworks

This section presents tools used for development and testing during the project.

2.2.1 Numpy

NumPy is a package for scientific computing in Python [49]. It provides support for large, multi-dimensional arrays and matrices, along with several mathematical and logical operations on these. Most arrays are stored in a more efficient way than Python lists and operations on them are faster to perform.

2.2.2 TensorFlow

TensorFlow is an open source library for performing numerical computations [1]. Although it can be used for computations in general, it is most commonly used as a tool for machine learning research. TensorFlow can be interfaced using Python and is then translated to a computational graph. By launching a TensorFlow session, data shaped as tensors, which is a generalization of N-dimensional arrays, can be fed into the computational graph. The graph performs a series of mathematical operations on the data. Weight matrices and biases are trainable variables in the TensorFlow graph during a session. Loss functions and optimization algorithms for backpropagation exist in TensorFlow. That makes training a model is as simple as specifying an objective function to optimize for, as well as running the optimizer with a batch of data inside a session.

2.2.3 Keras

Keras is a neural networks API for Python [10]. It runs on top of TensorFlow [1], CNTK [57] or Theano [2]. Keras is user-friendly and allows for complex models to be created with relatively few lines of code. Keras consists of many commonly used building blocks of neural networks. These are parts as layers, objectives, activation functions and optimizers. The components include parts for convolutional and recurrent neural networks as convolutions, pooling, dropout and batch normalization.

2.2.4 Jupyter Notebook

Jupyter Notebook is a web application for editing and running code [35]. Jupyter is open-source and can be used with over 40 different programming languages. A Jupyter Notebook document is a JSON document, and by following a standard, it can contain an ordered set of cells where each can contain code, text, mathematics, plots or other media.

2.2.5 ITK-SNAP

ITK-SNAP is a tool for visualizing 3D images and segmentations [69]. The tool is free, open-source, and multi-platform. As well as displaying mask files on top of images it allows for manual segmentation in three planes at once. A screen-shot from ITK-SNAP visualizing a 3D segmentation is shown in Figure 2.4.

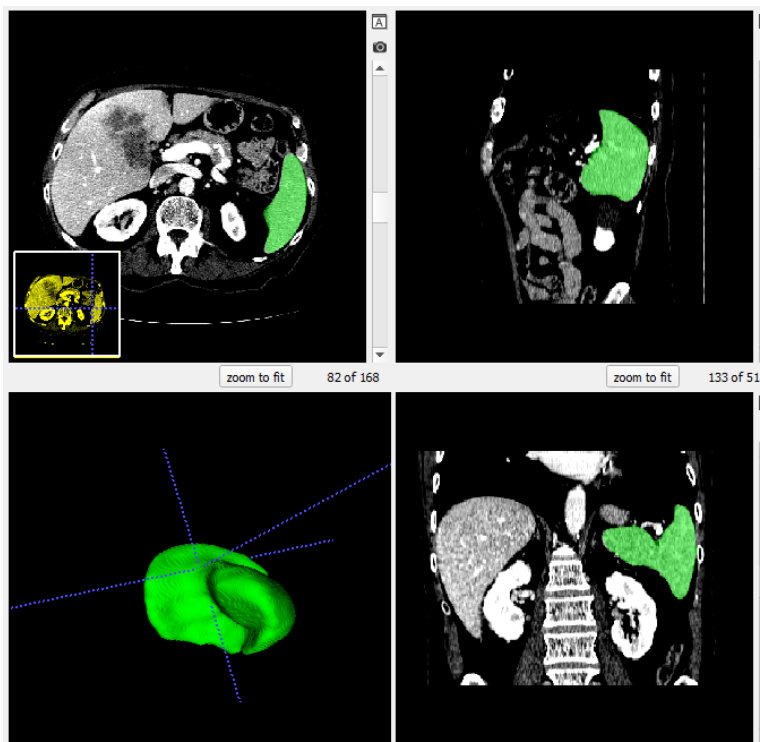


Figure 2.4: An example of 3D visualizations of a medical image in ITK-SNAP.

2.3 Computer Vision

Computer vision has the goal of creating computation models which can gain a high-level understanding of digital images or videos. The models should be able to perform tasks similar to what the human visual system can accomplish. This involves extracting, analyzing and understanding of information from images or video. A paper from 2015 showed how a computer vision model was able to classify images more accurately and way faster than human beings [53]. Computer vision models can be used to support, or even replace, humans.

An image is represented as a grid of numbers by a computer. Each number represents a pixel colour. To get a higher level of understanding of the image the computer needs to interpret these numbers. Two images of the same object usually consist of completely different pixel values. This makes it hard for a computer to determine if two images contain the same object by only looking at the pixel intensities.

In recent years computer vision has seen a *Deep Learning revolution*, driven by

the increase in computational power and the huge amounts of data available [17]. Moving from engineered image features to using deep neural networks has been a paradigm shift. Today, tasks like feature extraction, image classification, object detection, and image segmentation are usually solved by using deep neural networks.

2.3.1 Feature Extraction

The goal of feature extraction is to reduce an image representation into a set of values containing relevant, non-redundant information. The most discriminating information should be found in the extracted features. Depending on the task, some parts of the image might not be relevant and is thus not preserved in any features. By reducing the size of the image representation, less computational power and memory is required when processing it further. Predictions will also be more accurate when irrelevant information has been removed. Because of this, feature extraction is an important part of computer vision tasks.

2.3.2 Image Classification

Image classification is the task of assigning a label from a set of predefined classes to an image. An image classifier is trained using supervised learning by being supplied with a set of images, and their labels: e.g car, cat or human. The classifier must learn to extract features that capture the relationship between images and their labels.

2.3.3 Object Detection

Locating semantic objects in an image is called object detection. It involves both finding the location and determining the class of the objects. The location of each object is usually not defined accurately, but rather indicated by a box, as seen in Figure 2.5. Object detection is more complex than image classification, as it also needs to indicate the position of the object in question. In addition to indicating the location, an object detector distinguishes between instances of the same class.

2.3.4 Image Segmentation

Image segmentation aims to partition an image into several segments. While object detection locates a rectangle in which an object is present, image segmentation accurately indicates class membership by finding pixels belonging to the class. One application of image segmentation is, for example, to determine the exact position and size of a tumour in the brain. Two common image segmentation tasks

are semantic segmentation and instance segmentation.

In semantic segmentation, every pixel in an image is given a class. An example of segmentation is shown in Figure 2.5. Semantic segmentation does not distinguish between instances of the same class. The other type of segmentation is instance segmentation, which gives a segmentation mask for each instance of objects from the predefined classes. Indicating instances of the same class by different segmentation masks make instance segmentation a harder task than semantic segmentation.

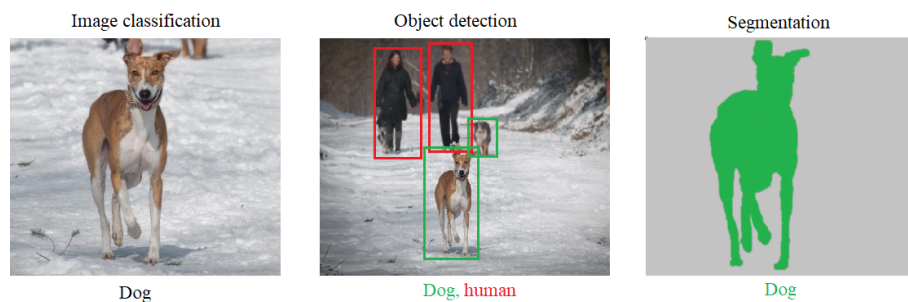


Figure 2.5: An illustration of common computer vision tasks; image classification, object detection and image segmentation.

Source: <https://www.pexels.com/photo/winter-dog-pet-dogs-104329/>

2.4 Artificial Neural Networks

Artificial neural networks (ANN) are mathematical models inspired by biological neural networks [25]. These networks are supervised, meaning that they learn a model from examples. An ANN consist of several layers, where each layer has a number of artificial neurons. The first layer is often referred to as the input layer, the last layer is the output layer and layers in between are called hidden layers.

An ANN is built up of artificial neurons. The idea of an artificial neuron was first explored by McCulloch & Pitts in 1943 [47]. It takes several inputs and transforms them by weights, bias and an activation function into an output. More specifically, the sum of the inputs is multiplied by the weights. Further, the bias is added and the result is passed through the activation function. The activation function maps the output and decides whether the neuron is activated or not. Different activation functions exist for different purposes. The weights and the bias of each neuron in an ANN must be learnt.

Random initial values are often used for weights and biases in an ANN. Further,

training samples are sent through the network one by one. For each sample, the output value is calculated. This process is called the forward pass. A loss function is used to compare the output value to the target. To minimize the error of the network, the weights are updated in the negative direction of the gradient of the loss function. This is done by propagating errors backwards through the network, updating the weights, in the process called backpropagation [42].

2.4.1 Weight Update

There are several ways to perform a weight update. In stochastic gradient descent (SGD), the weights are updated after each training sample is sent through the network [6]. Batch gradient descent updates the weights by using the error of all the training samples. This requires fewer calculations than SGD and is less sensitive to noise. A third way is mini-batch gradient descent. The weight updates are performed using the error over a subset of the training samples. It requires fewer weight updates than SGD and is more likely to get out of local minima, as the process is more stochastic than when using all training samples in batch gradient descent.

2.4.2 Activation Functions

An activation function can be applied to the output of neurons in a neural network. Some common activation functions are Sigmoid, ReLU, leaky ReLU and Softmax, see Table 2.2. In addition, a recently developed CapsNet Squashing function is listed. The Sigmoid function has traditionally been frequently used for activation. The advantage is that it is non-linear and scales the input to a range of zero to one. However, when using many layers the gradients tend to become very small, known as *the vanishing gradient problem*. This makes the weight update, and thus speed of learning, very slow. A function that does not have this issue is ReLU. It is a commonly used activation function today. ReLU adjusts all inputs below zero into zero and leaves higher values unchanged. That makes both the function and its derivative monotonic, which makes the network converge easier. One issue with ReLU is that a lot of values are mapped to zero, causing neurons to become inactive and they might not be able to ever activate again. A solution to this issue is to use leaky ReLU, which maps values below zero to a small, non-zero value. The parameter α determines the slope of the curve for negative values, where the default often is $\alpha = 0.1$. Another common activation function is Softmax. It maps the inputs into a probability distribution. It is therefore especially suitable at the end layer of a classification task. The squashing function used by capsule networks squashes a vector to a length below zero, having the same orientation as the original vector. The non-linearity is used in combination with a

dynamic routing algorithm.

Activation function	Formula
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$
ReLU	$f(x) = \max(0, x)$
Leaky ReLU	$f(x, \alpha) = \max(\alpha x, x)$
Softmax	$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$
CapsNet Squashing Function	$f(\vec{x}) = \frac{\ \vec{x}\ ^2}{1+\ \vec{x}\ ^2} \frac{\vec{x}}{\ \vec{x}\ }$

Table 2.2: Commonly used activation functions.

2.4.3 Loss Functions

A loss function is used for updating the weights of the network during the training process. After the forward pass, an error is calculated during backpropagation. This error is used by the loss function for updating the weights in the negative direction of the loss function’s derivative. Cross-entropy loss is commonly applied for classification problems, while Dice loss often is used for classification problems regarding medical segmentation. Spread loss was recently applied to capsule networks for image classification [26].

Weighted cross-entropy

Weighted cross-entropy weights the classes based on the fraction of the respective class in the total dataset. Thus, a class with a low fraction of the pixels in the dataset will get a high weighting. This is particularly interesting when the dataset contains unbalanced classes. The function is usually referred to as weighted binary cross-entropy when there is only one target class, making the total number of classes two. The formula is shown in Equation 2.1 for p probability of class, n classes and class weight w .

$$Loss = - \sum_i^n w_i * y_i * \log p_i \tag{2.1}$$

Dice loss

The values of Dice ranges from zero to one, where zero indicates no overlap and one indicates complete overlap between two segmentations. For two overlapping regions, the Dice is defined as two times the intersection over the union. Dice loss is defined in Equation 2.2 for k pixels, prediction p and label y .

$$Loss = 1 - \frac{2 \sum_k y_k * p_k}{\sum_k y_k^2 + \sum_k p_k^2} \quad (2.2)$$

repeated for all classes and averaged

There exist two variants of the Dice coefficient, where the difference is in the denominator. The Jaccard Dice loss squares the output and the target in the denominator, as in Equation 2.2, while Sørensen Dice loss does not.

Spread loss

A spread loss will directly penalize incorrect predictions that are closer to the activation of the positive class than a given margin. The penalty, in that case, is the square distance from the prediction to the margin m , for all n classes, except the positive class. This is shown in Equation 2.3.

$$Loss = \sum_{i, i \neq t}^n \max(0, m - (y_t - y_i)^2) \quad (2.3)$$

2.4.4 Learning Rate

Another important aspect of neural networks is how much the weights should be adjusted during backpropagation. The parameter controlling the amount of weight update is called the learning rate. If the learning rate is too small, reaching the minimum of the loss function might take a long time, or the optimizer might even get stuck in a local minimum. On the other hand, having a too large learning rate might make it impossible to reach the global minimum. It is therefore important to find a suitable value for the learning rate, depending on the task at hand. For being able to converge fast along with reaching a minimum loss, the learning rate is often decreased during training.

2.4.5 Adam Optimizer

Adam is an optimization algorithm first introduced in the paper *Adam: A Method for Stochastic Optimization* by Kingma and Ba [34]. The algorithm combines the

classical stochastic gradient descent method with two techniques that are commonly used by other optimization algorithms: momentum and adaptive per-parameter learning rates. Momentum calculates a weighted average of the gradients from previous training steps when estimating a new gradient in the network. This can help the network to escape local optima. While traditional stochastic descent has a single global learning rate used by all parameters in the network, Adam uses an adaptive learning rate for every parameter in the network. Per-parameter learning rates improve learning, particularly on problems with sparse gradients.

The optimizer has four hyperparameters which will influence how well it can learn a task. The parameters are:

1. *Alpha*: Initial learning rate
2. *Beta_1*: Exponential decay rate for the moment estimates (moment 1)
3. *Beta_2*: Exponential decay rate for the moment estimates (moment 2)
4. *Epsilon*: A small constant to prevent division by zero

Although the choice of hyperparameters will affect training, the algorithm is not very sensitive to changes in them. The parameters suggested for image classification problems in the paper are an *Alpha* of 0.001, a *Beta_1* of 0.9, a *Beta_2* of 0.999 and an *Epsilon* of 10^{-8} . This is claimed to work on many problems with sparse gradients.

2.5 Generalization and Regularization

To be able to perform well on unseen instances, a model must be able to generalize beyond the training data. A model is said to overfit the training data when there exists an alternative model that performs worse on the training set but better on the test set. Overfitting occurs because the model has enough flexibility to adapt to the noise found in the training data. To prevent overfitting and improve generalization several regularization techniques exist. Among these are early stopping [9], dropout [61], weight decay [37] and data augmentation.

2.5.1 Early Stopping

Early stopping is a technique where some of the available data is used as a validation set. This data is excluded from the training process. The error of the training set and the validation set are regularly calculated, and training is terminated as soon as the validation set error increases by a certain amount. The validation set

error should indicate how well the model is generalizing beyond the training data. When this error starts to increase, it is reasonable to assume that the algorithm is overfitting on the training data.

2.5.2 Dropout

When using dropout a random subset of the hidden nodes is disabled during training. The output from the remaining hidden nodes is scaled to compensate for the reduced neural activity. At each training step, a new subset is excluded. The amount of exclusion is regulated by the dropout rate. This technique forces the algorithm to not depend on outputs from single units, and thus improves the network's ability to generalize.

2.5.3 Weight Decay

Weight decay is often used to prevent the weights from fitting the training set too well. This is done by adding a regularization term to the loss function. The most commonly used terms are L2 and L1. L2 is the sum of the squared weights while L1 is the sum of the absolute value of the weights. These have different properties such as that L1 often produces sparse weight matrices and that L2 has analytical solutions which make the calculation more computationally efficient.

2.5.4 Input Reconstruction

Input reconstruction is concerned with recreating the original input data, at the output of a neural network. By training a network to reconstruct the input, it should learn how to make compressed representations of structures in the data, throughout the network. The objective of input reconstruction is commonly used as a regularization method in neural networks.

2.5.5 Data Augmentation

Generalization can also be improved by increasing the amount of training data. More data provides more information about the target model. A way to increase the training set is by data augmentation. In data augmentation, more data is created by transformations on the training set, but without altering the meaning of the data. Examples of transformations on an image can be flipping, rotating and adding noise. While some augmentation increases the robustness of the algorithm, too much or irrelevant transformations might make the training slow and/or the task hard to learn. Due to this, choosing transformations and the amount of

augmentation carefully is important. The augmentation techniques called elastic deformation and salt and pepper noise might not be self-explanatory and are presented in more depth.

Elastic deformation

Capturing and creating labelled segmentations for datasets of medical images require a lot of work. Most datasets available are therefore too narrow to learn the task of segmentation without any data augmentation. In the paper *Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis*, Simard et al. propose a way of expanding datasets through elastic distortions [58]. This technique works by generating a random displacement field of vectors for all pixels in a given image. The displacement field is convolved with a Gaussian smoothing filter having a standard deviation σ . When applying the processed displacement field to a training image, an algorithm such as bilinear interpolation is used to determine the final colour value for every pixel in the output image. In Figure 2.6, various displacement fields are applied to the MNIST dataset. By generating a large number of displacement fields, the same training observation can be augmented into many new, yet slightly different training examples.

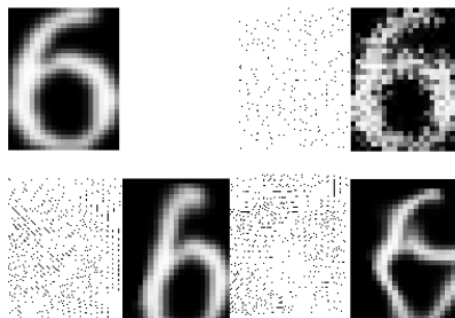


Figure 2.6: Elastic deformation on the MNIST dataset. Top left shows original image, while right and bottom shows pairs of displacement fields with various smoothing and resulting images.

Source: <http://cognitivemedium.com/assets/rmnist/Simard.pdf>

Salt and pepper noise

Salt and pepper noise is characterized by a sharp change in intensity for individual pixels in an image (See Figure 2.7). Noise can be artificially created for an image by changing a percentage of all pixels in the image to a random value. By adding salt and pepper noise to images during training, an image classifier will be more robust to this type of noise when presented in real images.



Figure 2.7: An example of salt and pepper noise in an image.

Source: https://upload.wikimedia.org/wikipedia/commons/f/f4/Noise_salt_and_pepper.png

2.6 Convolutional Neural Networks

A convolutional neural network (CNN) is a deep neural network, carefully designed for image recognition tasks. It was modelled after a study by D.H Hubel and T.N Wiesel on the monkey striate cortex [28]. They discovered that the visual cortex consists of receptive fields that detect light in overlapping subregions. In CNNs every neuron responds to stimuli in a restricted region, such that the overlapping regions of the neurons together cover the entire image. This approach overcomes the scalability problem that arises when regular neural networks are applied to image processing tasks.

The deep network consists of different types of layers including convolution layers, non-linear layers, pooling layers, and fully connected layers. An example of the structure can be seen in Figure 2.8. Each neuron of a convolution layer processes the pixels of their receptive field. The processing is done by element-wise multiplication between the field and a filter. The filter is applied to all pixels in the image, where the number of pixels between each application of the filter is called the stride. Thus, selecting a higher stride reduces the size of the output of convolution. To capture information in the boundaries of an image and/or avoid the image from shrinking, padding is commonly used. Padding is applied by adding a value outside the boundaries of the image. Commonly used paddings are zero-padding and same paddings. While zero-padding assigns zeros outside boundaries, a same-padding repeats the value found at the edge.

In a traditional convolutional network, a ReLU operation is often used in between the other layers. ReLU provides non-linearity and helps control overfitting by transforming negative values to zero. Pooling layers are used to reduce computational complexity and control overfitting. In a pooling layer, the input from several neurons is combined into one output. One common type of pooling is max pooling, which reduces a set of inputs into the maximum value. Fully connected layers are commonly used as the last layers of a convolutional neural network. Each neuron of a fully connected layer has weights to all the neurons in the previous layer. As the last layer is where the class is decided, this makes all the previous neurons involved in calculating the final output.

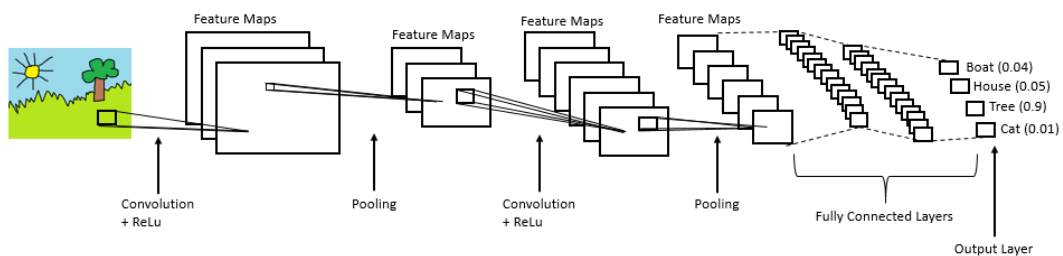


Figure 2.8: An example of a convolutional neural network.

Source: https://www.cs.cmu.edu/~rsalakhu/talks/talk_JSM_part4.pdf

2.6.1 AlexNet

AlexNet is a CNN that won the ImageNet Challenge in 2012, where it significantly outperformed its competitors [36][55]. The network was designed for image classification and consisted of eight layers. It used convolutions, ReLU activations, max pooling, dropout, SGD with momentum and data augmentation.

2.6.2 GoogLeNet

GoogLeNet was the winner of the ImageNet Challenge in 2014 [64]. It achieved the highest performance for both image classification and image detection. The network was inspired by LeNet but was wider and deeper, using 22 layers [39]. The network is built up of repeated modules called *Inception*. Each such module separately performs 1x1 convolution, 3x3 convolution, 5x5 convolution and a 3x3 max pooling on the input. Dimensionality reduction is achieved by applying 1x1 convolution before 3x3 and 5x5 convolution, as well as after the pooling operation. The four results are then concatenated by using a filter.

2.6.3 VGGNet

VGGNet was developed by the Visual Geometry Group from the University of Oxford [59]. It was placed 2nd in the ImageNet Challenge for image classification and won the localization task in 2014. The network consists of 16 convolutional layers using 3x3 convolutions and lots of filters. All hidden layers use ReLU, and the final prediction is made by applying three fully connected layers at the end.

2.6.4 Residual Neural Networks

A residual neural network is an ANN that uses residual blocks [24]. Residual blocks have connections that skip some layers, see Figure 2.9. One motivation behind this method was the difficulty faced when training deep neural networks. When a network grows deeper, the magnitude of the gradients in the earlier layers becomes small and learning becomes slow. Residual Neural Networks make it possible to grow networks deeper, while still being able to efficiently learn models. The ResNet architecture won the 2015 ImageNet Challenge for localization and detection, beating human-level performance.

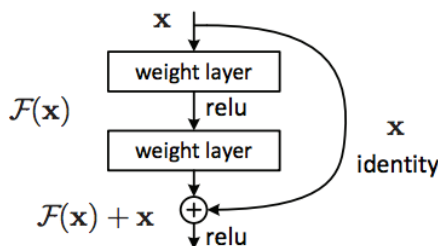


Figure 2.9: A residual block.

Source: <https://arxiv.org/pdf/1512.03385.pdf>

2.7 Image Segmentation Architectures

Following the success of convolutional neural networks for image classification tasks, they were also extended to performing segmentation. An efficient solution to the task was shown when the fully convolutional network (FCN) was introduced in 2014. Since this, many variants of CNN for segmentation has been developed.

2.7.1 Fully Convolutional Networks

The fully convolutional network (FCN) is a CNN architecture purposed by Long et al. in 2014 [43]. It was built on traditional image classification models such as

AlexNet [36], VGGNet [59], GoogLeNet [64] and ResNet [24], but is designed for semantic segmentation rather than classification. The architecture replaces fully connected layers with convolutional layers. This makes it output spatial maps instead of classification scores. The network takes inputs of arbitrary size and produces an output of the same size. This is done by a downsampling path followed by upsampling using deconvolutions. The FCN architecture was the first to show how CNNs could be efficiently trained end-to-end for a segmentation task.

2.7.2 SegNet

SegNet is an encoder-decoder network and has an architecture similar to FCN. SegNet was developed by Badrinarayanan et al. for semantic segmentation [5]. Like FCN, the architecture has a downsampling path and an upsampling path but uses max unpooling rather than deconvolutions in the upsampling path. This eliminates the need for the network to learn the upsampling and provides a more efficient way to achieve segmentations than FCN.

2.7.3 U-Net

U-Net is a fully convolutional neural network commonly used for biomedical image segmentation [54]. It was developed by Ronneberger et al. in 2015. The U-Net architecture has a contracting path that reduces the resolution of the encoded image, and an expanding path which later increases the resolution of the image, see Figure 2.10. The idea of reducing the resolution and forcing it through a bottleneck before its resolution is increased resembles the traditional auto-encoder.

By concatenating features from layers in the contracting path with the upsampled image in the expansive path, the network can create an accurate segmentation. The contracting path of the network has stacks of two convolutional layers and a max pooling operation, which is repeated until the desired resolution is achieved. The expanding path mirrors the contracting path, but performs upsampling and concatenations instead of max pooling, to increase the resolution.

U-Net has previously been shown to give state of the art performance on problems such as brain tumour segmentation [30]. Extensions to the original U-Net include the use of 3D volumes, instead of single 2D slices. Neighbouring slices in a 3D volume often have similar or shared features, which often are interesting to consider together when performing segmentation.

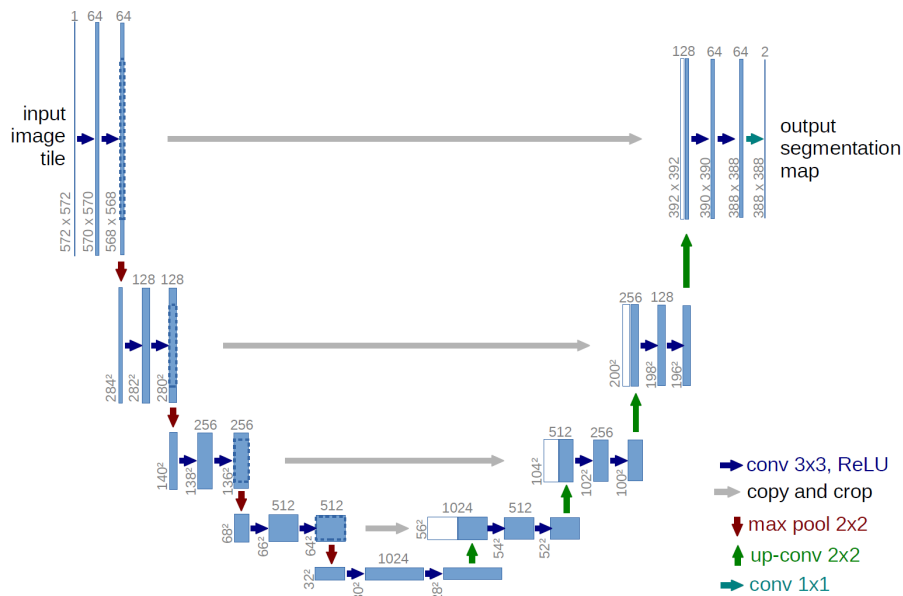


Figure 2.10: The U-Net architecture as presented in the original paper.
 Source: <https://arxiv.org/pdf/1505.04597.pdf>

2.7.4 Mask R-CNN

Mask R-CNN was developed by He et al. in 2017 [23]. It expands the Faster R-CNN architecture for object detection into an architecture for instance segmentation [52]. This architecture is built up of two main stages. First, a Region Proposal Network suggests bounding boxes of interest [19]. The second stage uses Fast R-CNN for classifying them and producing a segmentation mask. Mask R-CNN minimizes the error of both the bounding box proposal and the segmentation mask. When published, the architecture showed state-of-the-art results on instance segmentation and human pose estimation.

2.8 Capsule Networks

Traditional convolutional neural networks perform exceedingly well at recognizing the presence of patterns in images by using learned filters. The output from a convolutional filter that is applied to a region in an image is a single scalar value, where a high value indicates that it is likely that a given feature is present. This process is typically repeated several times with pooling layers in between. Max

pooling is used for routing information between layers.

An example is a convolutional neural network, which is trained to detect faces. If an image of a face is presented to the network, neurons will activate, each indicating if for example an eye, a nose or a mouth was detected. If all these features that are typically parts of a face are present, a face is detected in the next layer. This process of assigning parts to wholes has worked surprisingly well in practice but is inherently flawed. The network may still believe that an image is a face even though the relations between the eyes, nose, and mouth are wrong, as illustrated in Figure 2.11. CNNs use routing by max pooling and retain lots of contextual information, but lacks information about the spatial relationship between features. In the end, the network is left with the knowledge of what is likely in the image, but not their instantiation parameters; i.e. lighting, texture, deformation, position, orientation, etc.



Figure 2.11: An example where a CNN often would incorrectly classify the image as a face.

Source: <https://towardsdatascience.com/capsule-networks-the-new-deep-learning-network-bd917e6818e8>

Capsule networks attempt to solve this by introducing the concept of capsules. Capsules are groups of neurons and the building blocks of capsule networks. A single capsule is responsible for detecting the presence of an entity in an image, as well as a limited subset of its most relevant instantiation parameters. An entity can, for example, be a nose with a given rotation, skew, and deformation. In the case of detecting a face, capsule networks would only assume that an image contains a face if the eyes, nose, and mouth have a correct relation. This learned relationship would typically include the position relative to each other.

Capsule networks solve the problem of assigning parts to wholes using the process of dynamic routing. The dynamic routing algorithm finds a set of coefficients that controls how much information is forwarded from every capsule in a lower layer to every capsule in a higher layer. These coefficients are not trainable parameters in the network, but rather parameters that are calculated in every forward pass for every input. The trainable parameters in capsule networks are transformation matrices that transform capsule outputs into output predictions of the capsules in the next layer. This routing scheme is a lot more sophisticated than how CNNs with max-pooling routes information by only selecting the highest activated feature. Figure 2.12 shows how a capsule network could detect that even though all parts of a face are present, their position, size, and orientation do not match.

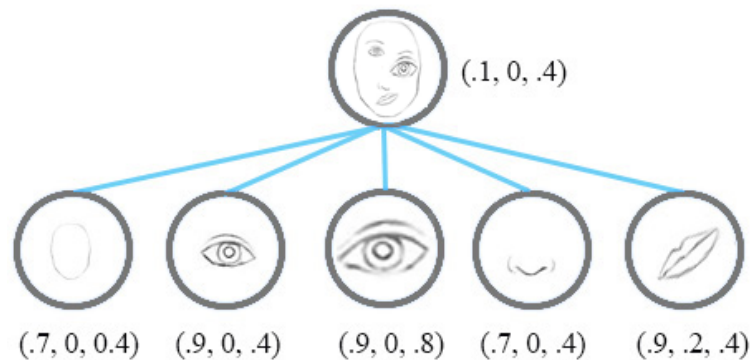


Figure 2.12: The parent capsule in this capsule network would detect that the parts of the face are in disagreement. It gives the image a low probability of being a face.
Source: <https://towardsdatascience.com/capsule-networks-the-new-deep-learning-network-bd917e6818e8>

2.8.1 Dynamic Routing

The first recognized type of capsule networks was presented in the paper *Dynamic Routing between Capsules* by Sabour et al. [56]. In this implementation of capsule networks, every capsule outputs a vector with a fixed dimension. The dimension of a capsule tells how many convolutional filters it will learn, and effectively limits the number of parameters the capsule can learn about entities. While performing face detection, an object could be any type of edge, a corner, a nose, a mouth, or even the face itself. An entity is defined as an instance of an object, having certain properties. The entity could, for example, be 25° rotated along the first axis and translated 10 pixels up, relative to the capsules reference point. An example

of how a parent capsule with several child capsules might look like is shown in Figure 2.13.

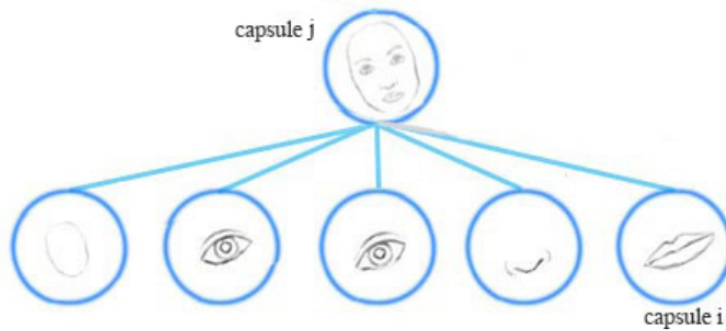


Figure 2.13: An example of a parent capsule with several child capsules in agreement.

Source: <https://jhui.github.io/2017/11/03/Dynamic-Routing-Between-Capsules/>

The output of a capsule is a condensed vector, where each element in the vector ideally represents one of the parameters used to instantiate the entity. The Euclidean length of the output vector represents the likelihood of an entity, with its instantiation parameters, being present. Conceptually, as long as the magnitude of the capsule's output vector remains the same, it will detect the same entity with equal probability, independent of the direction the capsule vector points in. An interesting observation is that by moving the capsule vector along the hypersphere, it will detect an entity with the same probability, but have completely different instantiation parameters. If a capsule is used to reconstruct the input image, one can, in theory, observe which instantiation parameters every dimension of the capsule attempts to model, by changing the latent variables of the capsule vector in different directions. An example of input reconstruction after modifying some of the dimensions of the output capsule is shown in Figure 2.14.




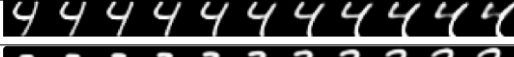


Scale and thickness	
Localized part	
Stroke thickness	
Localized skew	
Width and translation	
Localized part	

Figure 2.14: Changing the latent variables of the capsule vector effects how images of numbers from the MNIST dataset are reconstructed. Each row shows how the reconstruction is affected by moving along a single dimension of the output capsule.

Source: Dynamic Routing between Capsules [56]

All capsules send their predictions to the next layer of capsules in a coordinated manner using a dynamic routing algorithm. The goal of dynamic routing is to determine how a child capsule's output should be distributed among its parents. Before the dynamic routing starts, the predicted output vectors for all parent capsules are calculated by every child capsule. Equation 2.4 shows how the output predictions $u_{\hat{j}|i}$ are calculated from every child to every parent capsule, using the child capsule output u_i and a trainable transformation matrix W_{ij} .

$$u_{\hat{j}|i} = W_{ij}u_i \quad (2.4)$$

During dynamic routing, the coupling coefficient c_{ij} between every child and every parent capsule is determined. A coupling coefficient can be thought of as the probability of a parent entity being present given the child entity. The coupling coefficients of a child is a probability distribution summing to one. To compute the coupling coefficients c_i for a child capsule, log likelihood probabilities b_i are estimated using an iterative process. Usually, the log likelihood priors, b_0 are initially set to 0.

In every iteration, the dot product between a child capsule's prediction for its parent $u_{\hat{j}|i}$ and the actual output of the parent \hat{v}_j is calculated. The dot product is a measure of agreement between them and is treated as log likelihood estimates. For several iterations, usually in the range of 3-5, the log likelihoods are refined by:

- (i) Calculating the coupling coefficients c_i by performing Softmax on the current set of estimated log probabilities b_i for every capsule i in layer l . See Equation 2.5.

-
- (ii) Calculating the weighted sum of prediction vectors from layer l : s_j , using the previously calculated coupling coefficients for every capsule j in layer $(l+1)$. See Equation 2.6.
 - (iii) Calculating the capsule output v_j by squashing the predictions using a non-linearity that scales the length of the output vector to less than 1, but leaves its orientation unchanged, for every capsule j in layer $(l+1)$. See Equation 2.7.
 - (iv) Updating the log likelihood estimates by adding the dot product of child capsule predictions $\hat{u}_{j|i}$ and the parent capsule output v_j . See Equation 2.8.

After the appropriate number of iterations, the final capsule output v_j is returned for all capsules in layer $(l+1)$.

$$c_i = \text{softmax}(b_i) \quad (2.5)$$

$$s_j = \sum_i c_{ij} \hat{u}_{j|i} \quad (2.6)$$

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|} \quad (2.7)$$

$$b_{ij} = b_{ij} + \hat{u}_{j|i} \cdot v_j \quad (2.8)$$

This routing scheme works in theory because lower level capsules in agreement will contribute towards a similar higher level capsule output. Predictions in agreement will have vectors pointing in the same direction; giving a high value for the dot product between them. If two predictions are in disagreement, their vectors will point in different directions giving a lower dot product between them. When many child capsules agree on an output prediction, their predictions will be reinforced with a higher dot product which gives a higher contribution to its log probability estimates. When using multiple iterations of this process, the predictions in most agreement will be stronger and stronger in a positive feedback loop.

2.8.2 Matrix Capsules With EM-Routing

Matrix capsules solve the dynamic routing problem in a very different way than the routing algorithm in *Dynamic Routing between Capsules*. Instead of calculating a vector that tries to encode the instantiation parameters of entities, a matrix

capsule attempts to encode the pose of an entity as a 4x4 matrix. In addition to estimating the pose, it also detects the entity activation explicitly as a scalar, similarly to how a neuron in a CNN detects the presence of a feature. The pose matrix will ideally be able to capture information about the instantiation parameters of an entity, while the activation scalar provides a level of confidence whether or not the detected entity is the same object as the capsule attempts to model.

The pose and the activation of an entity detected by a capsule are modelled by a single Gaussian distribution. The distribution attempts to cluster the pose prediction of a parent capsule to agree with lower level capsules. The *Expectation-maximization algorithm* (EM) variant is used iteratively to estimate the mean and standard deviation of the distribution that best fits the child capsule predictions, as well as estimating the routing coefficient controlling the level of influence by each child capsule.

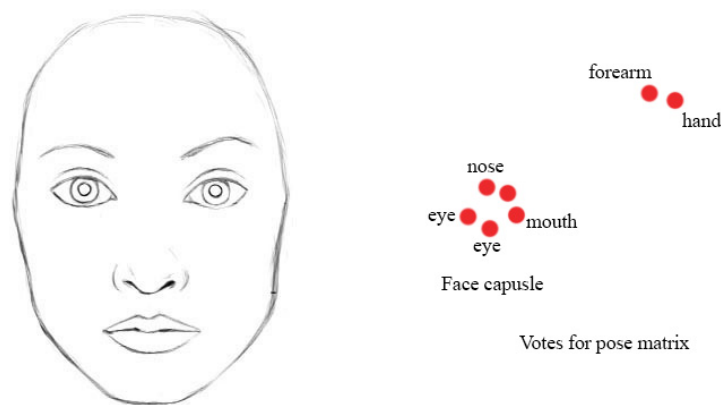


Figure 2.15: Simplified view of how EM clustering of capsule predictions could look. The clustering happens in a 16-dimensional space in practice.

Source: <https://jhui.github.io/2017/11/14/Matrix-Capsules-with-EM-routing-Capsule-Network>

With the EM clustering method, predictions in disagreement with the majority will be located further away from the cluster centre and have a higher cost of activation. This is illustrated in Figure 2.15. The cost of activating a child capsule using a parent capsule is dependent on the part-whole relationship between the child capsule and the parent. The probability of a child capsule being activated by a parent capsule is dependent on the distance to its cluster centre. Seeing that the cluster centre is decided by the majority vote of other capsules, it explains how the presence of related features will justify the activation of a higher level capsule that represents an entity composed of those features. The latent transformation matrix

that is used to calculate a prediction based on the pose of a child capsule, is a weight matrix learned using backpropagation. A pose prediction V of a capsule i is the matrix convolution of its pose matrix M_i and the weight matrix W , which is shared between all the capsules in the same layer. This is shown in Equation 2.9. These matrix convolutions are not the same as regular convolutions. In regular convolutions, the sum of the element-wise product between a convolution kernel and the input image is calculated. Matrix convolutions, on the other hand, multiplies each element of the kernel as a transformation matrix by the pose matrix at a given location in the receptive field.

$$V_i = M_i W \quad (2.9)$$

The vector form of V has $4 \times 4 = 16$ dimensions in total and constitutes the dimensionality of the Gaussian distribution to estimate. The expression in Equation 2.10 shows the probability density function that the h 'th component of a child capsule prediction is drawn from the same h 'th component of the Gaussian distribution represented by a parent capsule.

The cost of explaining the prediction cast by the child capsule i is the sum over all h components of the negative log probability density $-\ln(P_{i|j}^h)$. See Equation 2.11 and Equation 2.12.

$$P_{i|j}^h = \frac{1}{\sqrt{2\pi(\sigma_j^h)^2}} \exp\left(-\frac{(V_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2}\right) \quad (2.10)$$

$$\ln(P_{i|j}^h) = -\frac{(V_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2} - \ln(\sigma_j^h) - \frac{\ln(2\pi)}{2} \quad (2.11)$$

$$cost_{i|j} = r_{ij} \sum_h -\ln(P_{i|j}^h) \quad (2.12)$$

The cost of activating the h 'th component of a parent capsule j , is found by calculating the weighted sum the costs going from every child capsule to a given parent capsule, as shown in Equation 2.13. To give the capsule a probability in range from 0 to 1, it is put through an activation function (Equation 2.14).

$$\begin{aligned}
cost_j^h &= \sum_i (-r_{ij} \ln(P_{ij}^h)) \\
&= \frac{\sum_i r_{ij} (V_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2} + (\ln(\sigma_j^h) + \frac{\ln(2\pi)}{2}) \sum_i r_{ij} \\
&= (\ln(\sigma_j^h) + \frac{1 + \ln(2\pi)}{2}) \sum_i r_{ij} \\
&= (\ln(\sigma_j^h) + \beta_u) \sum_i r_{ij}
\end{aligned} \tag{2.13}$$

$$a_j = \text{sigmoid}(\lambda(\beta_a - \beta_u \sum_i r_{ij} - \sum_h cost_j^h)) \tag{2.14}$$

Pseudo-code for the EM-routing algorithm is shown in Listing 2.1. The algorithm receives activations from capsules in the previous layer, and receives the predictions calculated by multiplying poses from the previous layer with a pose-invariant transformation matrix. The number of iterations is typically in the range of three to five.

algorithm EM-Routing (a=activations ,V=predictions , iterations):

```

forall  $i \in \Omega_L, j \in \Omega_{L+1} : R_{ij} = \frac{1}{|\Omega_{L+1}|}$ 
for  $it$  in  $\text{range}(1, \text{iterations} + 1)$ :
     $\lambda = \frac{1 + (\text{iterations} - 1) * it}{\max(1.0, \text{iterations} - 1.0)}$ 
    forall  $j \in \Omega_{L+1} : \text{maximization\_step}(a, R, V, j, \lambda)$ 
    forall  $i \in \Omega_L : \text{expectation\_step}(\mu, \sigma, a, V, i)$ 

```

```

poses =  $\mu$ .reshape( $|\Omega_{L+1}|, 4, 4$ )
return a, poses

```

function maximization_step(a,R,V,j , λ):

```

forall  $i \in \Omega_L : R_{ij} = R_{ij} * a_i$ 
forall  $h : \mu_j^h = \frac{\sum_i R_{ij} V_{ij}^h}{\sum_i R_{ij}}$ 
forall  $h : (\sigma_j^h)^2 = \frac{\sum_i R_{ij} (V_{ij}^h - \mu_j^h)^2}{\sum_i R_{ij}}$ 
forall  $h : cost^h$  calculated with Equation 2.13.
 $a_j$  calculated with Equation 2.14

```

function expectation_step(μ, σ, a, V, i):

```

forall  $j \in \Omega_{L+1} : p_j = \frac{1}{\sqrt{\prod_h 2\pi(\sigma_j^h)^2}} \exp(-\sum_h \frac{(V_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2})$ 

```

$$\forall j \in \Omega_{L+1} : R_{ij} = \frac{a_j p_j}{\sum_{k \in \Omega_{L+1}} a_k p_k}$$

Listing 2.1: An algorithm sketching the steps taken in the EM-routing algorithm. Ω_L is the set of lower level child capsules in layer L, and Ω_{L+1} is the set of higher level parent capsules. β_a and β_u are trainable parameters. λ is the inverse temperature which increases for every routing iteration. The algorithm receives a set of activations and predictions from child capsules in layer L, and returns the activations and pose matrices for layer L+1.

The network is trained using a spread loss starting with a small margin of 0.2, which is increased linearly during training to 0.9. By starting with a small margin value, the issue of *dead capsules* is prevented in the earlier layers due to high punishment of incorrect predictions.

When using convolutions to extract features, the same filter is applied to different regions of the image. To compensate for spatial information loss between layers, a technique known as *coordinate addition* is used. This technique adds the scaled coordinates from the centre of the current receptive field to the two first elements of the prediction matrix. This will encourage the transformation matrices to encode the position relative to the capsule’s receptive field in its predictions.

2.8.3 Segmentation

SegCaps extends the idea of capsules from the paper *Dynamic Routing between Capsules* into an architecture performing semantic image segmentation. All layers in SegCaps are convolutional and similar to the U-Net architecture. Further, SegCaps contains a contracting path and an expansion path with capsule concatenations between them. A consequence of the concatenation is that capsules in the contracting path also learn to predict the outputs of capsules in the expansion path. In order to reduce image size in the contracting path, some of the convolutional capsules have strides of two, halving the dimensionality in all axes. Similarly, to increase the image resolution, SegCaps uses deconvolutional capsules performing strided transposed convolutions instead of regular convolution.

The differences between the routing algorithm used for SegCaps and the original Capsule network implementation are subtle. Contrary to the original capsule network, SegCaps has no fully connected layers. When using convolutional capsules, weights are shared among capsules at different locations in the image, in the form of a convolutional kernel. The paper *Capsules for Object Segmentation* refer to this variant of the dynamic routing algorithm as the *Locally-Constrained Dynamic Routing* algorithm, but in practice, it works the same way as the convolutional capsules in the original paper.

Using weight sharing and locally-constrained routing, the number of trainable parameters in SegCaps is reduced significantly allowing for segmentation of images up to 512x512 pixels in size. The network is trained using a combination of weighted cross-entropy loss and an MSE reconstruction loss. The reconstruction loss attempts to reconstruct the pixel intensities of the pixels belonging to the target class. While the purpose of optimizing the weighted cross-entropy is to create accurate segmentations, the reconstruction loss is applied for regularization of the model. By forcing the network to learn how to reconstruct relevant parts of the input image, it is expected that the capsules better will learn meaningful representations of structures in the data.

2.9 Previous Work

The U-Net architecture was introduced by Ronneberger et al. in the paper *Convolutional Networks for Biomedical Image Segmentation* [54]. The architecture is fully convolutional and contains an encoder and a decoder path. Features taken from specific locations in the encoder path is concatenated with feature maps from specific points in the decoder path. Using this technique, the U-Net architecture is capable of creating pixel accurate segmentation. The implementation was tested on the task of segmenting neuronal structures in electron microscopic recordings and cell segmentation, as a part of the ISBI cell tracking challenge 2014 [45]. The model achieved very good results overall, and achieved above state of the art results on the challenge with a large margin.

In the paper *Brain Tumour Segmentation and Radiomics Survival Prediction: Contribution to the BRATS 2017 Challenge*, Isensee et al. presented a U-Net based model used for brain tumour segmentation [30]. The architecture was residual-based and accepted three-dimensional image volumes of 128^3 voxels. The implementation placed third in the BraTS 2017 competition [48], and achieved overall good Dice scores of 0.896, 0.797 and 0.732 for the whole tumour, tumour core and enhancing tumour, respectively. The authors concluded that the proposed architecture is effective at segmenting medical image datasets with few images, without suffering the consequences of overfitting. The paper acknowledges that efficient augmentation methods such as elastic deformation are important for preventing overfitting.

The winning submission of the Medical Image Decathlon was the framework *nnU-Net*, short for no-new U-Net, presented in the paper *nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation* by Isensee et al. [31]. The submitted implementation consisted of an ensemble of three simple variants

of the original U-Net: Cascade U-Net using downsampled images, 2D U-Net, and 3D U-Net. The paper emphasizes the importance of model adaptation, preprocessing, training and postprocessing. By using a sophisticated way of generating image patches of appropriate size and adapting the network topology according to the data that is used, the framework gave very good results at the challenge. The submission placed first for all seven tasks in the first phase of the competition, except for edema segmentation in the brain tumour challenge (Table 2.3), and was placed first for all tasks except spleen in the second phase (Table 2.4).

	BraTS			Heart	Liver		Hippoc.		Prostate	Lung	Pancreas		
	1	2	3	1	1	2	1	2	1	2	1	2	
Results	67.7	47.7	68.2	92.8	95.2	73.7	90.4	89.0	75.8	89.6	69.2	79.5	52.3

Table 2.3: Results from nnU-Net when tested on data from phase one of the Medical Segmentation Decathlon. Numbers reported are the Dice scores in percent for the different label classes in every dataset.

	Hepatic Vessel		Spleen	Colon
	1	2	1	1
Results	63	69	96	56

Table 2.4: Results from nnU-net when tested on data from phase two of the Medical Segmentation Decathlon. Numbers reported are the Dice scores in percent for the different label classes in every dataset.

Sabour et al. presented a type of artificial neural network called Capsule networks, along with an algorithm for training them in the paper *Dynamic Routing between Capsules* [56]. The paper challenges some of the inherent limitations of convolutional neural networks. Information is typically routed between convolutional layers using max-pooling layers, which results in loss of spatial relations between features. The suggested solution is to use a sophisticated dynamic routing algorithm in combination with capsules. The capsules themselves attempt to capture information about the instantiation parameters of an entity, and the routing algorithm decides how much information that should be routed from every capsule in a capsule layer to every capsule in the next. The paper showed that the capsule network architecture was capable of achieving above state of the art performance on the task of classifying highly overlapping digits using the MNIST dataset [40].

The authors concluded that capsule networks are promising in the field of computer vision, but still require a lot of work before they can be applied to images or volumes of a larger size than MNIST.

Building on the work from *Dynamic Routing between Capsules*, the same group of researchers released a second paper called *Matrix Capsules with EM Routing* in 2018 [26]. The paper suggests a new approach to the dynamic routing problem using the Expectation-maximization algorithm. This approach improves the first capsule network implementation in several ways. Firstly, the new architecture does not use the squashed capsule vector length to determine activation. Secondly, it does not use the dot product of two vectors to measure agreement, but rather a negative log variance of a Gaussian cluster. Because cosine saturates at 1, the dot product is not the best choice for differentiating between good and very good predictions. Lastly, the EM-routing method uses a matrix to represent the pose of an entity, rather than a vector. This reduces the number of parameters needed in the transformation matrix significantly. The capsule architecture proposed gave state of the art performance at detecting objects at novel viewpoints using the small-NORB dataset [41].

In 2018 a paper named *Capsule Networks against Medical Imaging Data Challenges* looked at the usefulness of capsule networks for medical image applications [32]. The paper primarily looked at classification problems, and applied capsule networks to the MNIST challenge [40], the Fashion-MNIST challenge [67] along with TUPAC16 (mitosis detection) [65] and DIARETDB1 (diabetic retinopathy detection) [33] for medical classification. The authors compared capsule network to LeNet and showed that CapsNet, in almost all cases, outperformed LeNet when having a limited amount of training data. The same conclusion is made for experiments with a class imbalance. The last experiment showed how SegCaps also outperforms LeNet when adding data augmentation. The conclusion states that CapsNet requires less training data, is robust to class imbalance and at the same time achieves the same, or better, performance than convolutional nets. The authors reported a training time of 1-3 minutes per epoch for CapsNet, while LeNet used 1-2 seconds per epoch, both depending on the number of classes to be predicted. Despite a significantly higher training time for CapsNet, the paper recommends exploring it further for medical image challenges, and in particular medical segmentation tasks, due to the good performance when having a high class-imbalance.

The first architecture for semantic image segmentation using capsule networks was reported in the paper *Capsules for Object Segmentation* from 2018 [38]. The network is much deeper in terms of layers than the original capsule network by

Sabour et al. while having roughly the same number of trainable parameters. The increase in complexity without increasing the number of trainable parameters is a consequence of not using any fully connected layers, but rather using only convolutional and deconvolutional capsule layers. Instead of forwarding every child capsule to every parent capsule at every location, the predictions are forwarded to parent capsules using a shared kernel. These modifications made it possible to segment images up to 512x512 pixels, which also gave competitive results at the task of lung segmentation using images from the LUNA16 dataset. The results are shown in Table 2.5.

Method	Parameters	Split-0	Split-1	Split-2	Split-3	Average
U-Net	31.0M	98.353	98.432	98.476	98.510	98.449
Tiramisu	2.3M	98.394	98.358	98.543	98.339	98.410
Baseline Caps	1.7M	82.287	79.939	95.121	83.608	83.424
SegCaps (R1)	1.4M	98.471	98.444	98.401	98.362	98.419
SegCaps (R3)	1.4M	98.499	98.523	98.455	98.474	98.479

Table 2.5: The performance of SegCaps with 1 and 3 routing iterations compared to regular U-Net, Tiramisu and a regular capsule network in the SegCaps paper. Numbers are reported in percentages of Dice score.

The paper *Fully CapsNet for Semantic Segmentation* shows how a modified version of capsule networks [63] outperforms fully convolutional networks on the PASCAL VOC dataset [16]. The architecture modifies the dynamic routing algorithm of Capsule networks into using partial connections instead of full connections. This reduces the complexity in terms of required memory and computations. The measurements were made using accuracy on the segmentation of objects from PASCAL VOC such as airplane, sofa, bird, and boat. Further, their experiments show that Fully CapsNet significantly outperforms FCN on the segmentation of slightly rotated objects. However, the authors mention that the high memory usage and computation requirements are some things that need to be handled in future work.

Trace-back along Capsules and its Application to Semantic Segmentation is a paper published in 2019 addressing the use of Capsule Networks for segmentation [63]. The network purposed is called Tr-CapsNet and contains an operation sequence which derives class memberships of each pixel by recursively capturing part-whole relationships. The architecture has 3 modules. The feature extraction module consists of several convolutional layers. The next module is called *the capsule and traceback module* and contains capsule layers with a traceback

pipeline followed by one convolutional layer. At the end, the upsampling module increases the size of the signal for the model to output a prediction of the intended size. Deconvolutions are used for upsampling. The proposed architecture was tested on the modified MNIST dataset [40] and a hippocampus dataset from the Alzheimer’s Disease Neuroimaging Initiative (ADNI). The performance was measured using pixel accuracy, mean class accuracy, and Dice score. A comparison to a state of the art U-Net was made. For both datasets, the authors reported slightly improved results using Tr-CapsNet compared to U-Net.

Chapter 3

Methodology

This chapter describes the methodology used for exploring the research questions. It introduces the architectures, the datasets and the preprocessing used, the conducted experiments, the training procedures and how the results were evaluated.

3.1 Architectures

Four architectures were used in the experiments. The 2.5D U-Net and the EM-SegCaps architectures were implemented solely for this project. The SegCaps architecture was the exact same as the one developed and published by LaLonde et al. [38]. The Multi-SegCaps architecture was a modification of SegCaps, where the changes were implemented during the course of this work.

3.1.1 2.5D U-Net

The implementation of 2.5D U-Net follows the architecture used by Isensee et al. in the paper *Brain Tumour Segmentation and Radiomics Survival Prediction: Contribution to the BRATS 2017 Challenge* [30] closely. However, it uses 2D convolutions instead of 3D convolutions and spatial dropout in three dimensions was replaced with spatial dropout in two dimensions. Due to the architecture being fully convolutional, it is capable of processing input slices of arbitrary size, having an arbitrary number of input channels and an arbitrary number of output classes. For a visual representation, see Figure 3.1.

The U-Net variant has a contracting path which reduces the image resolution, an expansion path that increases the image resolution and concatenation operations that combines features from the contracting path with features from the expansion path. Convolutions in the contracting path are surrounded by residual skip con-

nections.

The contracting path consists of several context modules, where each has two 3x3 2D convolutions with a spatial dropout 2D layer in the middle. The dropout probability was 0.3. The first convolution of a context module uses a leaky ReLU activation, while the second convolution has no activation. The context module is a residual block, which means that the output of the context module is element-wise summed with its input. The residual sum is then activated using Leaky ReLU. Afterwards, the activated feature maps are forwarded to the next layer in the contracting path, as well as saved for concatenation in the expansion path. Throughout the network, leaky ReLU uses a slope $\alpha = 0.01$.

To reduce the computational burden of performing repeated convolutions on high-resolution images, the resolution of feature maps are halved in all axes at the start of every layer in the contracting path. This is accomplished using a 3x3 convolution with a stride of two. The convolution outputs twice as many feature maps as it receives. This is done to avoid losing valuable context information, resulting from a loss in resolution. The output from this convolution is forwarded to the next context module, and the process is repeated four more times. The contracting path is in total five layers deep, starting with 16 convolution filters in the first layer and ending with 256 filters in the last layer.

At each level of the expansion path, feature maps are forwarded to an upsampling module, which first upscales them by a factor of two using nearest neighbour interpolation, before a 3x3 convolution is applied to reduce the number of filters in half. The upsampled features are then concatenated with features from the corresponding level of the contracting path before they are forwarded to a localization module. In the localization module, features from the concatenation are recombined with a 3x3 convolution, which is followed by a 1x1 convolution that halves the number of features again. The output from the last convolution is forwarded to the next upsampling module, and the process repeats itself three more times. At the top layer of the expansion path, the localization module is replaced with a regular 3x3 convolution.

The output from the 3x3 convolution in the highest layer of the expansion path, along with the output from the localization modules at the two levels below are forwarded to segmentation layers. A segmentation layer is simply a 1x1 convolution, which outputs the same number of filters as there are classes in the dataset. The segmentation from all three levels is combined in a specific way. The lowest level segmentation is first upsampled using nearest-neighbour interpolation before it is element-wise added to the output of the segmentation layer for the next level.

The residual sum of the two segmentation layers is then upsampled once more before it is element-wise summed with the output of the last segmentation layer. This is done to accelerate convergence time. The last element-wise sum is the output of the network. No activation is applied to the output, which differs from the original paper. The reason why this was done was to avoid saturation of softmax, which can happen with very imbalanced datasets.

3.1.2 SegCaps

The neural network architecture *SegCaps* presented in the paper *Capsules for Object Segmentation* by LaLonde et al., was used to perform binary segmentation of medical images. The code for the architecture was obtained from a GitHub repository they had created¹. The architecture used is called SegCaps R3, referring to the use of three iterations of the dynamic routing algorithm in each capsule. The exact architecture is given in Figure 3.2.

A regular 2D convolution is performed on an input image to produce 16 feature maps. The tensor consisting of 16 feature maps is reshaped in order to give it a new dimension of length one, such that the reshaped tensor now represents a single 16D capsule. The 16D capsule is forwarded to the primary capsule layer, which is a regular convolutional capsule layer with one routing iteration, which returns predictions of two 16D capsules.

Following the primary capsule layer, there are sets of two convolutional capsule layers at every layer for the remainder of the contracting path. The first of the two operations is a 5x5 convolutional capsule layer with stride two, which also doubles the number of feature maps that are output. The second layer consists of 5x5 convolutional capsules without a stride. Output features from the latter operation are forwarded to the next level of the contracting path, as well as stored for concatenation with capsules in the expanding path.

After predictions are downsampled in the contracting path, they are then upsampled in three levels of the expanding path. Every level of the expanding path contains two capsule layers. The first layer is a 4x4 deconvolutional capsule, which upsamples the image by a factor of two using transposed convolution. The output vectors from deconvolutional capsules are then concatenated with capsule output from the corresponding level in the contracting path before they are used to predict the second layer of capsules. The second capsule layer uses the concatenated predictions from both the contracting and expanding path, to predict a layer of

¹<https://github.com/lalonderodney/SegCaps>

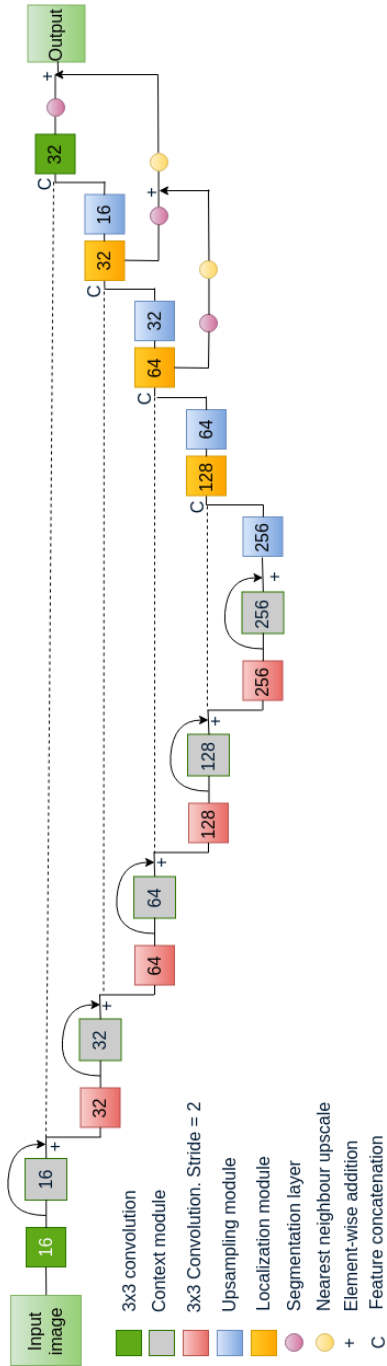


Figure 3.1: A diagram of the 2.5D U-Net architecture.

5x5 convolutional capsules. Instead of using 5x5 convolutional capsules in the last layer, they are replaced with 1x1 convolutional capsules, which produces the final segmentation vectors. The segmentation vectors are converted into actual predictions by calculating the Euclidean length of them. For each pixel in the image, it is classified as the target class if its length is longer than a given threshold.

To perform reconstruction of the pixels belonging to the target class, the output from the last layer is reshaped from capsule form into a convolutional form. This involves flattening the 16D capsule into 16 feature maps. Using ground truth labels, the pixels that do not belong to the target class are masked out. This prevents the reconstruction head from considering irrelevant background pixels. Three layers of 2D convolutions with a 1x1 kernel is applied to the masked feature maps. The last of the three convolutions output a single filter giving a reconstruction of the target class of the input image. A mean square error loss between pixels from the input image and the reconstructed image is used in combination with the segmentation loss, to train the network. To train the segmentation part of the network, a weighted cross entropy loss is calculated using the Euclidean length of the output capsule as prediction and a ground truth label that is 0 for background and 1 for the target class.

3.1.3 Multi-SegCaps

The architecture proposed in the paper *Capsules for Object Segmentation* by LaLonde et al., was extended to support end-to-end segmentation of different datasets containing an arbitrary number of input modalities and output classes. Input modalities could be several types of images acquired of the same tissue, this is commonly obtained for MRI. Several output classes could, for example, be several parts of a brain tumour, which has different labels. Due to the support for multiple target classes, the architecture was called Multi-SegCaps.

The output capsule layer was modified to output N 16D output capsules, where N is the number of classes in the dataset, including background, and the predicted class is the one represented by the capsule with the longest euclidean length. The network is trained using the predicted capsule lengths and one-hot encoded ground truth labels.

Instead of only trying to reconstruct a single target class, multi-class SegCaps attempts to reconstruct the pixels belonging to all classes, except the background class. The segmentation capsule's output is masked with a ground truth mask for all these classes. The N 16D masked capsules are then flattened into $N * 16$ convolutional feature maps that are forwarded to the reconstruction head. The re-

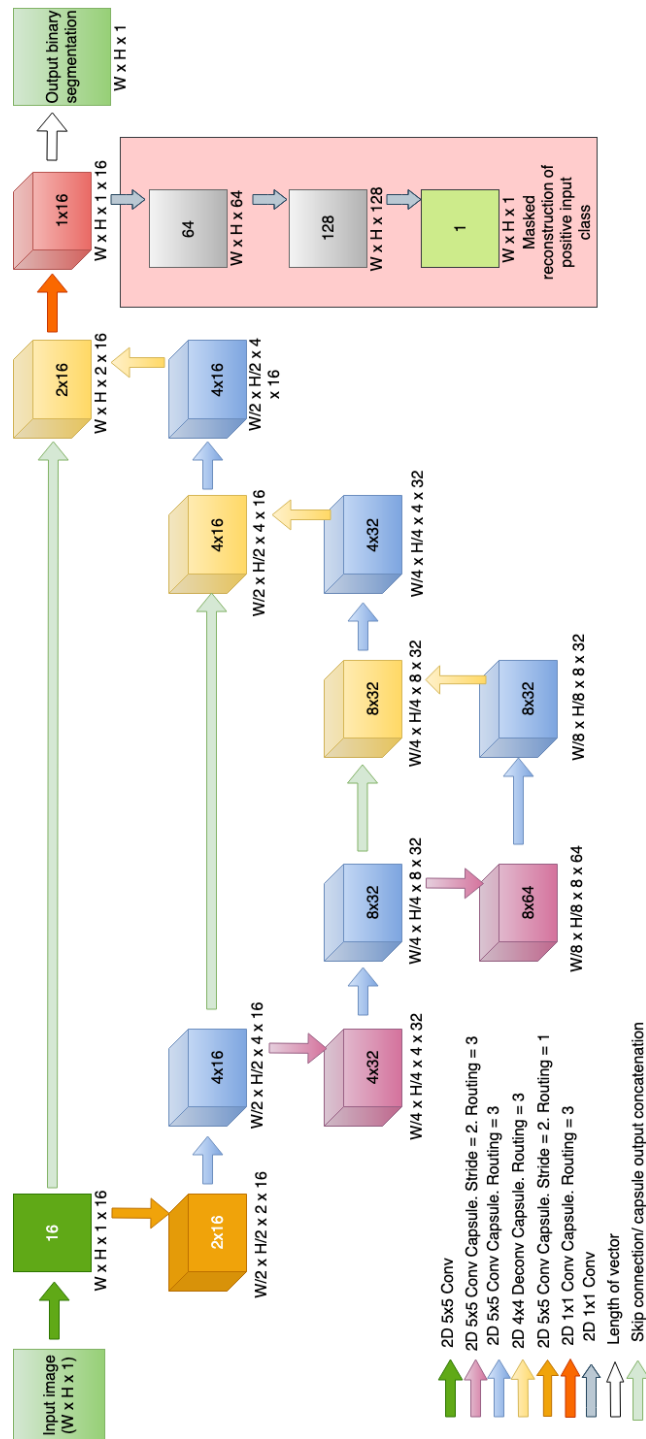


Figure 3.2: A diagram of the SegCaps architecture. The numbers in blocks show how many capsules with a given dimension that is outputted from the respective layer.

construction head consists of three convolutional layers, each with a kernel size of 1. The last of the three convolutional layers will output as many filters as there are input modalities. This incentivises the network to store properties about more than one input channel, if available. The total reconstruction loss is the mean squared error of pairwise output reconstruction and positive masked input modalities.

3.1.4 EM-SegCaps

The method from *Matrix Capsules With EM-Routing* was extended to perform semantic image segmentation. The network has a U-Net-style encoder-decoder topology, similar to the original SegCaps architecture, and is referred to as EM-SegCaps. The architecture uses matrix capsules with EM-routing and is shown in Figure 3.3.

A regular 2D convolution with kernel size 3x3 is applied to an input image, producing 16 feature maps that are forwarded to the primary capsule layer. The primary capsule layer is responsible for transforming the output from the first convolution into an estimated pose and activation of a single capsule.

The primary capsule layer performs two sets of 2D convolutions with a 1x1 kernel. The first convolution transforms the 16 input feature maps into 16 predictions for each receptive field. The 16 prediction values represent the current pose of the entity that is detected and can later be reshaped into a 4x4 matrix. The second convolution in the primary capsule layer is also given the same 16 feature maps as the previous convolution. The difference between them is that the second convolution only calculates a single capsule activation for each receptive field. In other words, for every pose matrix that is calculated, a single scalar is also detected. The scalar is treated as the activation of the capsule and works similarly to how convolutional kernels calculate activations in a convolutional neural network. The difference between the activation value of a capsule in a capsule network and a neuron in convolutional networks, is that capsule networks do not use the activation directly when calculating the output. It is rather used as a coefficient that influences the amount of information that is forwarded from a capsule to different capsules in the next layer. A sigmoid non-linearity is applied to the activation values in order to scale them between zero and one. The primary capsule layer concatenates the pose values and activation into 17D vectors, and returns it as its output.

Apart from the first primary capsule layer, the capsule network architecture also contains convolutional capsule layers, strided convolutional capsule layers, and deconvolutional capsule layers. Convolutional capsule layers accept the poses

and activations from capsules in the previous layer and output new poses and activations for the capsules in the next. This is accomplished using the *Expectation-maximization routing algorithm* (EM-routing). Before performing EM-routing, all child capsules cast an initial vote of the output for every capsule in the next layer, using its own pose matrices. Casting this vote involves multiplying the pose matrix by a trained transformation matrix going into the parent capsule, which is shared by all child capsules. This effectively translates into performing a modified version of 2D convolution. Instead of performing pointwise multiplication between a convolution kernel and the input image, each element in the kernel is a transformation matrix that is multiplied by the pose matrix located at a specific location in the receptive field. After the predictions are calculated, they are forwarded to the EM-routing algorithm, along with the activations from the previous layer. The EM-routing algorithm is run for three iterations before it returns the final pose and activations for all capsules in the current layer.

Strided convolutional capsules are almost identical to regular convolutional capsules. The only difference is that a stride of 2 is used during matrix convolution to reduce the feature maps in half along the height and width axes. Deconvolutional capsules are also similar to regular capsules, as they perform regular capsule convolution after upscaling by a factor of two, using nearest-neighbour interpolation. They are technically not deconvolutional capsules as they don't perform transposed convolution, due to the technical challenges of implementing transposed matrix convolutions.

At the time of writing Tensorflow has not implemented every operation needed to build this architecture. By rolling out the convolution operation as a single matrix, which repeats the pixel values according to how they would be multiplied by a convolutional kernel, it was possible to emulate the operation using Python and simple matrix multiplication in Tensorflow. However, this approach is very demanding in terms of memory consumption. The EM-routing algorithm does have a low-level Tensorflow implementation either but is rather written in Python, which combines a series of simpler Tensorflow operations to accomplish its goal.

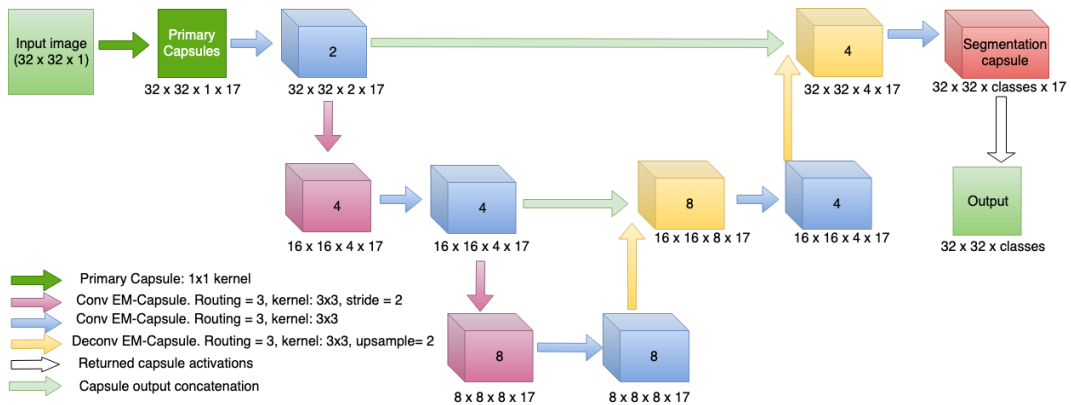


Figure 3.3: A diagram of the EM-SegCaps architecture.

3.2 Datasets

Seven datasets were used in this project. The LUNA16 dataset was applied by LaLonde et al. in the paper *Capsules for Object Segmentation*. In this project, it was used when testing the reproducibility of their work. Multiple datasets from the collection Medical Segmentation Decathlon (MSD) were used to test the generalizability of the developed architectures on medical image segmentation tasks [60]. The collection contains the BraTS dataset, which also can be found as a separate challenge [48]. Several of the datasets from MSD were selected to test the 2.5D U-Net as well as for testing SegCaps, Multi-SegCaps, and EM-SegCaps. An overview of all datasets used in the experiments can be seen in Table 3.1. The test volumes were only used for 2.5D U-Net when submitting the test set segmentations to the MSD challenge.

Dataset	Modality	Training	Testing	Shape	Target classes
LUNA16	CT	888	-	512x512xZ	1
Spleen	CT	41	20	512x512xZ	1
Heart	MRI	20	10	320x320xZ	1
Hippocampus	MRI	263	131	XxYxZ	2
Liver	CT	131	70	512x512xZ	2
Pancreas	CT	282	139	512x512xZ	2
BraTS	Multimodal MRI	484	266	4x240x240x155	3

Table 3.1: The datasets used in the experiments. Where X, Y or Z means that the given dimension was of variable size.

3.2.1 LUNA16

The LUNA16 dataset for segmentation consists of 888 CT scans. The structure to be segmented is the lungs. Each volume has two dimensions of 512 pixels and a median volume depth of 237.5 slices. Originally the dataset consisted of several classes, but for binary classification, the left lung, the right lung, and lung nodules are all combined into one class. An illustration of how the volumes look like is shown in Figure 3.4.



Figure 3.4: The left image shows an example slice from the LUNA16 dataset, the centre image shows the same slice with its label and the right image shows a 3D visualization of the label from a different angle.

3.2.2 Spleen

The spleen dataset contains 61 CT images. Only 41 of the images has a given label and 20 volumes were reserved for testing. Like the LUNA16 dataset two dimensions are 512 pixels. The median volume depth is 90 slices. An example image is shown in Figure 3.5.

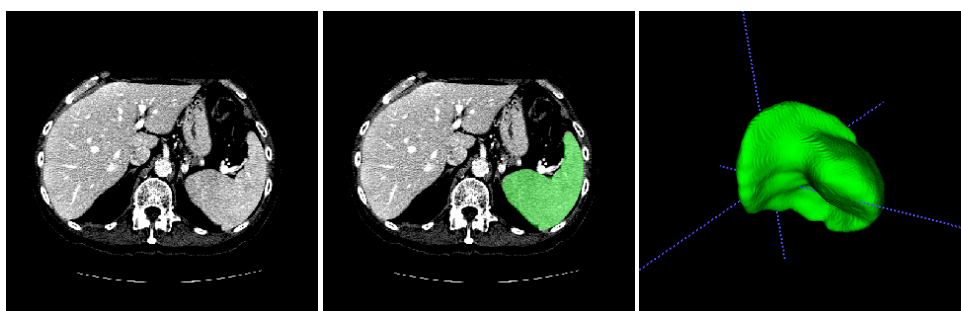


Figure 3.5: The left image shows an example slice from the spleen dataset, the centre image shows the same slice with its label and the right image shows a 3D visualization of the label from a different angle.

3.2.3 Heart

The heart dataset aims to segment the left atrium. An example can be seen in Figure 3.6. The dataset consists of only 20 MRI volumes for training. The slices from the volumes are 320x320 pixels in size and have a median depth of 115 slices. This task is challenging compared to the others because of the small training set and a significant class imbalance between left atrium and background.

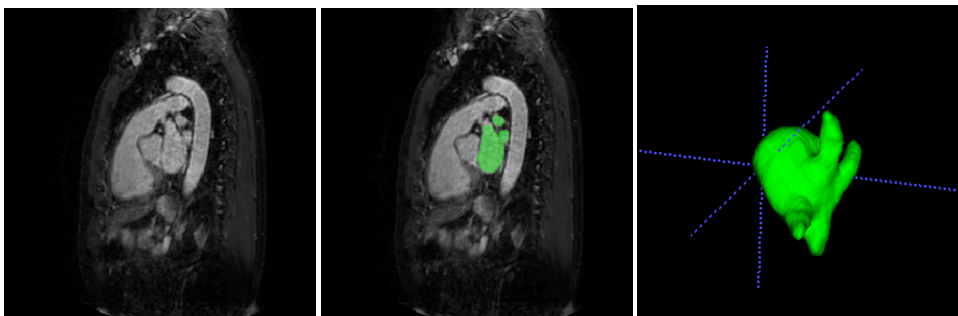


Figure 3.6: The left image shows an example slice from the heart dataset, the centre image shows the same slice with its label and the right image shows a 3D visualization of the label from a different angle.

3.2.4 Hippocampus

The hippocampus dataset has 263 training images. The images are from MRI scans and are in the range of 35-45 pixels in size, for all dimensions. Image slices were cropped to 32x32 pixels during training, for compatibility with network architectures. An overview of the method used for training and inference on this particular dataset is shown in Table 3.2. Because of the requirements of the architectures and the variable size of the hippocampus volumes, the volumes were cropped during training and padded during validation and testing. The dataset has two labels: posterior and anterior hippocampus, also known as hippocampus head and body. The small image sizes made this dataset suitable for experimenting with new segmentation architectures. An example slice, as well as an illustration of a 3D label, are shown in Figure 3.7.

Method	Training shape	Testing shape
2.5D U-Net	32x32	64x64
SegCaps	32x32	48x48
Multi-SegCaps	32x32	48x48
EM-Segcaps	32x32	48x48

Table 3.2: The input shapes used for training and validation/testing on the hippocampus dataset.

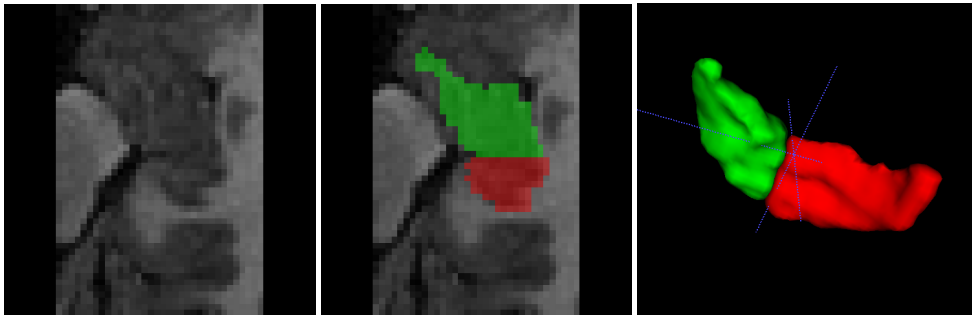


Figure 3.7: The left image shows an example slice from the hippocampus dataset, the centre image shows the same slice with its label and the right image shows a 3D visualization of the label from a different angle.

3.2.5 Liver

The main challenge of the liver dataset is to segment two neighbouring classes which are highly imbalanced; a large liver class and a small liver tumour class. The dataset has 131 CT volumes that are reserved for training. A slice from a liver volume is 512x512 pixels in size, while the depth of volumes varies a lot, ranging from around 100 to 1000 slices. The median depth of liver volumes is 432 slices. An example of how the volumes look like is shown in Figure 3.8.

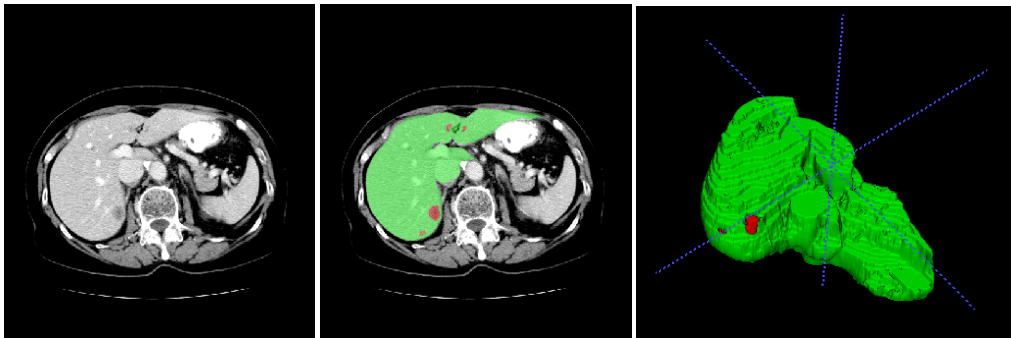


Figure 3.8: The left image shows an example slice from the liver dataset, the centre image shows the same slice with its label and the right image shows a 3D visualization of the label from a different angle.

3.2.6 Pancreas

The pancreas dataset consists of 281 labelled images and 139 test images. It has two labels where the pancreas is of medium size and the pancreas tumour is small. The structures are shown in Figure 3.9. The CT volumes are of 512x512 pixels with a median depth of 93 slices.

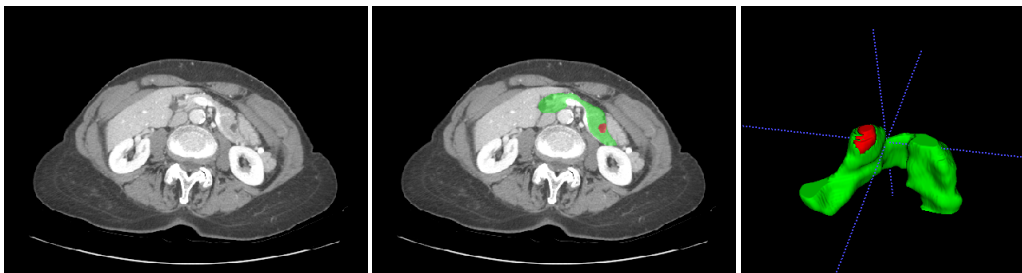


Figure 3.9: The left image shows an example slice from the pancreas dataset, the centre image shows the same slice with its label and the right image shows a 3D visualization of the label from a different angle.

3.2.7 BraTS

The BraTS 2017 dataset is a collection of 3D MRI scans from different patients. All images have four channels, each representing a different MRI imaging technique. Each modality has a dimension of 240x240x155 voxels. The modalities represented in the dataset are FLAIR, T1-weighted (T1w), T1-weighted with gadolinium contrast (T1Gd) and T2-weighted (T2w). 484 training volumes were used for this project.

Image labels have three classes: edema, non-enhancing tumour and enhancing tumour. As described by BraTS: "These tumour substructures meet specific radiological criteria and serve as identifiers for similarly-looking regions to be recognized through algorithms processing image information rather than offering a biological interpretation of the annotated image patterns" [48]. An example of the four modalities and three label classes are shown in Figure 3.10.

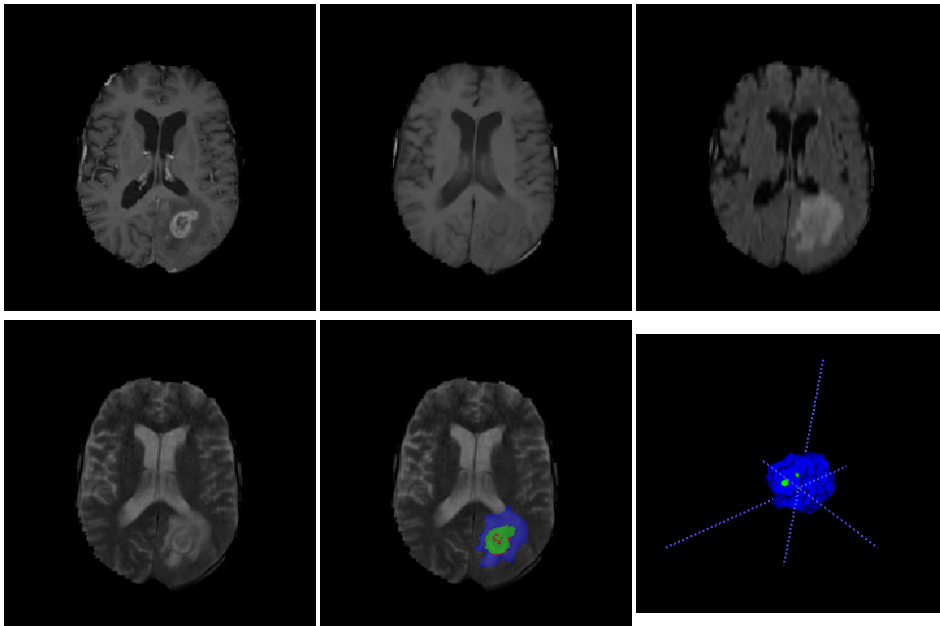


Figure 3.10: The top row shows three modalities of an example slice from the BraTS dataset, the bottom row shows the last modality, the same slice with its label and a 3D visualization of the label from a different angle.

3.3 Preprocessing

Normalization and augmentation were applied to images before training. While normalization was only done once, augmentation was applied repeatedly, with different parameters for every training iteration.

3.3.1 Normalization

The CT datasets were normalized and clipped using Hounsfield units with organ-specific level and window values. The LUNA16 dataset was normalized similar to how LaLonde et al. did in their work with SegCaps. The other datasets were normalized using their minimum and maximum values. The hippocampus dataset

had images with a variable intensity range and thus used a different normalization for each image. Each of those images was normalized by using the minimum and maximum value in the respective image. For the BraTS dataset, the minimum and maximum values were found for each modality in the order FLAIR, T1w, T1Gd, T2w. The values used for normalizing all datasets are shown in Table 3.3.

Dataset	Level	Window	Min	Max
Spleen	45	250	-80	170
Liver	60	250	-65	185
Pancreas	33	500	-217	283
LUNA16	-	-	-1024	3072
Heart	-	-	0	2196
Hippocampus	-	-	-	-
BraTS	-	-	0, 0, 0, 0	6476, 9751, 11737, 5337

Table 3.3: The values used for normalization of the datasets. The level and window are given for the CT data that were clipped and normalized by using Hounsfield units.

3.3.2 Augmentation

Augmentations were applied to the training data randomly. Table 3.4 shows the applied augmentation and the probability of applying each transformation to an image slice while training. An example of elastic deformations to the BraTS dataset can be seen in Figure 3.11. The other transformations are illustrated on a slice from the heart dataset in Figure 3.12.

Augmentation	Parameters	Probability
Rotation	max_degrees=45	10%
Flipping	axis=0	10%
Flipping	axis=1	10%
Shifting	max_horizontal=0.2, max_vertical=0.2	10%
Shearing	amount=16	10%
Zooming	max_zoom_range=(0.75, 0.75)	10%
Elastic deformations	alpha=1000, sigma=80, alpha_affine=50	20%
Salt and pepper noise	salt=0.2, amount=0.04	10%

Table 3.4: The different augmentations used in the experiment, their parameters and probability being applied.

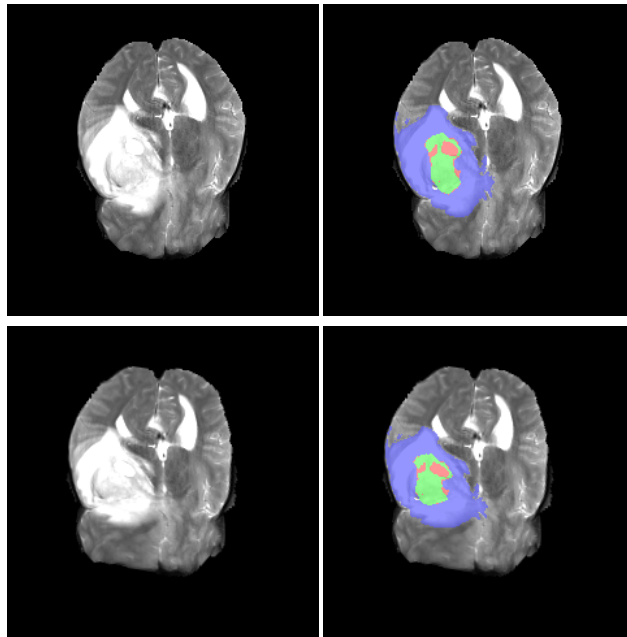


Figure 3.11: An example image with label on the first row and the same image after applying elastic deformations at the bottom row

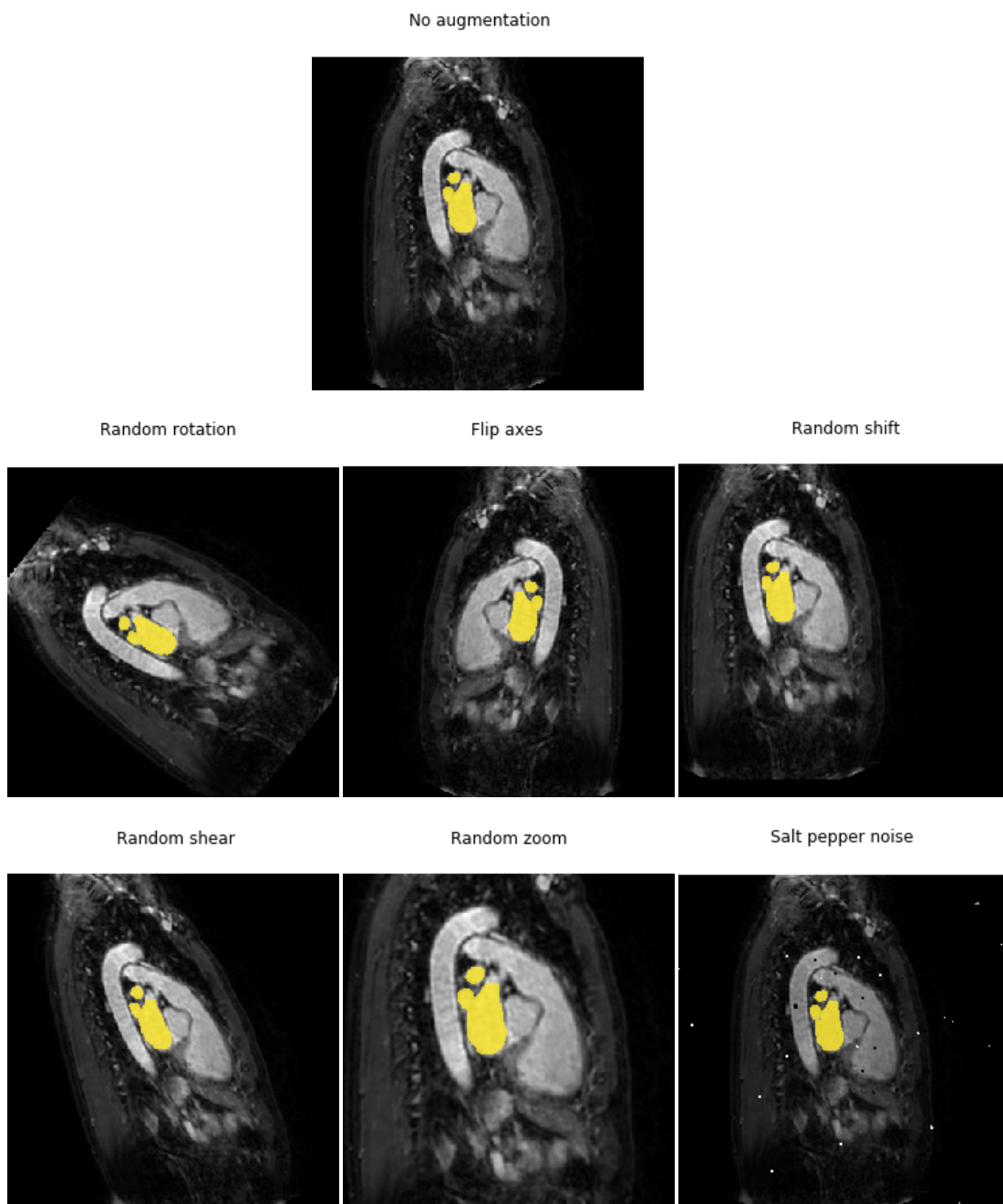


Figure 3.12: Augmentations used in the experiments. The augmentations are exaggerated for this visualization.

3.4 Experiments

Four experiments were conducted, one for each research question. The goals were to test the general performance of 2.5D U-Net, reproduce the results of SegCaps, as well as applying SegCaps to new datasets, explore a multi-class version of SegCaps and investigate the possibility of using EM-routing for SegCaps.

3.4.1 Experiment 1: 2.5D U-Net for Medical Segmentation Tasks

In order to address the first research question, **RQ1**, the developed 2.5D U-Net architecture was trained on various medical segmentation tasks. The motivation behind this experiment was to explore the possibilities of a general model that is capable of adapting to different types of medical segmentation problems without manual parameter tuning. It was also desirable to explore how well the architecture would perform compared to state of the art results. Due to this, the obtained results were submitted to the Medical Segmentation Decathlon.

The architecture was trained on six different datasets. The spleen dataset showed the performance on a dataset with a structure of a reasonable size and a homogeneous appearance. This dataset was also used for investigating the impact of using five consecutive slices as an input to the model rather than just a single slice. By training the model on the heart dataset the ability to generalize from only a few training samples was tested. The hippocampus dataset measured the performance on low-resolution images as well as the ability to learn two neighbouring labels with a similar appearance. The liver dataset was used to test the ability to learn two target classes where one of them is negligible compared to the other. By learning to segment the pancreas and pancreas tumour, the architecture was tested on a similar task as liver, but on smaller and more complex structures. The final evaluation on BraTS showed the ability to segment three target classes of tumour from multi-modal MRI volumes.

3.4.2 Experiment 2: Reproduce Results from *Capsules for Object Segmentation*

Applying the SegCaps architecture to three datasets would address the second research question, **RQ2**. The goal was to reproduce the results presented in the paper *Capsules for Object Segmentation*. Further, it was desired to test if the architecture would perform well on other datasets without modifications. In the original research paper, the results were given for the LUNA16 dataset and in this experiment, the same architecture and parameters were applied to test the validity

of the results reported by LaLonde et al.

The spleen and the heart dataset were used to test if SegCaps would be able to generalize to new datasets without modifying the network topology or parameters. The reason why the spleen dataset was a good fit for this experiment is that it is similar to LUNA16 in many ways. Both datasets contain CT images in the same intensity range, and both datasets have CT images with identical slice dimensions. Lastly, the target classes for both datasets contain a single class besides background with similar size structures: lungs and spleen. The heart dataset is of similar difficulty as the spleen dataset. It contains fewer images which make it faster to learn, but has more background voxels which result in a higher class imbalance, giving the network a different challenge.

3.4.3 Experiment 3: Multi-class Segmentation using SegCaps

Capsule networks have previously shown promising results at performing binary image segmentation with the SegCaps architecture. A lot of segmentation problems require that the algorithm used is capable of segmenting images into multiple classes. To test the feasibility of performing multi-label semantic segmentation, the SegCaps architecture was in this project modified to handle multiple classes.

The Multi-SegCaps model was trained on the spleen dataset, the heart dataset, the hippocampus dataset, the liver dataset, and the BraTS dataset. The spleen and heart datasets showed the performance impact of using a multi-class architecture on a problem which could otherwise be solved using a plain binary segmentation architecture. This would address the first part of **RQ3**, regarding the possibility of multi-class segmentation and whether multi-class SegCaps would be able to perform segmentation with the same performance as SegCaps. The first part of this experiment used the spleen dataset for comparing the loss functions Dice and weighted cross-entropy. The motivation behind this comparison was that Dice loss had shown good results on the 2.5D U-Net, while the original SegCaps used weighted binary cross-entropy. The hippocampus and liver dataset showed the performance of multi-SegCaps when learning two target classes. Lastly, the BraTS dataset was selected to test if the model was capable of handling this complex problem containing multi-modal MRI and as many as three target classes. By applying Multi-SegCaps to datasets with multiple target classes, **RQ3** regarding how well SegCaps adapts to multi-class segmentation was addressed further.

By using a similar training process as for 2.5D U-Net, the results for Multi-SegCaps would be comparable to the U-Net based model. This would give an

answer to the second part of **RQ3**, which concerned the performance of SegCaps compared to a U-Net based model.

3.4.4 Experiment 4: SegCaps with EM-routing

The dynamic routing algorithm proposed in the paper *Dynamic Routing between Capsules* by Sabour et. al, has gained a lot of publicity in recent years. The capsules architecture was originally designed to perform image classification but has later been adapted to multiple domains including segmentation. The same group of researchers publicized a second paper: *Matrix capsules with EM routing*, claiming that the proposed method might be able to solve the dynamic routing problem in a way that is better than the one in the original paper. However, the paper hasn't gained the same level of publicity yet. The authors suggest improving capsule networks by using matrix capsules and EM-routing.

In order to address **RQ4**, concerning the possibilities and limitations of using matrix capsules with EM-routing for segmentation, a new type of capsule network architecture was designed for this project. The architecture was trained to perform both binary segmentation of hippocampus, as well as multi-class segmentation of anterior and posterior hippocampus.

The encoder-decoder style EM-SegCaps network was compared with Multi-SegCaps on segmentation of hippocampus images. The experiment would show if SegCaps with EM-routing would be able to learn the tasks using only 30,166 trainable parameters, compared to the 1,436,769 parameters used by Multi-SegCaps.

3.5 Training

During training, batches were generated by drawing random 3D images from the pool of available data. Because all models use a type of 2D convolution, the images used for training were randomly sampled 2D slices from the 3D volumes. Slices from the images were sampled from the axis of the lowest resolution, which is referred to as the depth axis. A given image slice was normalized, augmented and then added to the batch if its label contained at least one pixel from a positive class. If the slice contained only pixels from the background class, it was simply discarded. The reason was to decrease the effect of class imbalance, as all datasets contained several times more of the background class than the target classes. The process of selecting image slices was repeated until a batch had the desired number of elements.

3.5.1 2.5D U-Net

The training process of 2.5D U-Net was similar for all datasets. The batch size was 24, one epoch had 1000 steps and training was ended when the validation score had not improved over the last 100 epochs. The model with the best validation score was used for testing. The initial learning rate was set to 0.001 and was reduced to 70% every time the model did not improve over 5 epochs. A $Beta_1$ parameter of the Adam optimizer was set to 0.9. During training, augmentation was used and five consecutive slices were used as input to the model. The exception was the first part of the experiment, which compared the performance of using one and five consecutive input slices. Dice loss was minimized while training.

3.5.2 SegCaps

The training process of SegCaps was similar to the one used by LaLonde et al. It lasted up to 200 epochs, but was terminated when the validation Dice score had not improved for 25 epochs. One epoch is 10000 training steps with a batch size of 1. Weighted cross-entropy loss and MSE reconstruction loss were minimized with equal weight. The Adam optimizer was used for stochastic gradient descent with an initial learning rate of 0.0001 and a $Beta_1$ of 0.99. The learning rate was decayed by a factor of 0.05 if the validation Dice score did not improve the last 5 epochs. Training data was normalized and augmented.

3.5.3 Multi-SegCaps

The training was performed using a batch size of 1, due to memory limitations. One epoch had 1000 steps, and the model early stopped if it did not improve for 100 epochs. These parameters were selected to match the ones used for 2.5D U-Net. After early stopping, the model with the best validation score was stored, and used for testing. For every validation, 500 steps were used. The initial learning rate was set to 0.0005 and was reduced to 70% every time the model did not improve over 5 epochs. The Adam $Beta_1$ parameter was set to 0.99, due to the low batch size. Augmentation and reconstruction weighting was used for regularization with a reconstruction weight of 0.01. 5 consecutive slices were used as input for making a prediction for the middle slice. For all experiments except the first one, which used Dice loss, the loss function used was weighted cross-entropy. An example of the calculated weights used for weighted cross-entropy for the spleen dataset can be seen in Table A.5.

3.5.4 EM-SegCaps

The EM-SegCaps architecture requires that all the images have the same dimensions during training, so every slice in all the hippocampus images and masks were cropped into 32x32x32 pixels. During inference, the hippocampus images were zero padded to fit entire image slices into a batch, but the padded voxels were removed before calculating performance metrics. Images were normalized and used augmentation.

A modified version of the multi-class Jaccard Dice loss was used for optimization. This loss function was used to reduce the effect of class imbalance. The original Dice loss function calculates a total Dice score for all classes. The modified version calculates separate Dice loss scores for all classes, which are then averaged into a single total Dice score.

3.6 Evaluation

Several evaluation metrics were used to determine the quality of the models. Precision, recall and the Dice similarity coefficient were used to compare the target segmentation mask and the predicted segmentation mask. The metrics were calculated in several ways, which are described in the second part of this section.

3.6.1 Metrics

Precision and recall are two commonly used metrics for evaluating how useful and complete a method is. Precision, also known as the positive predictive value, is the fraction of relevant cases compared to the retrieved cases. Recall, also known as the sensitivity, is the fraction of relevant retrieved cases compared to the total relevant cases.

A high precision could be achieved by classifying a few relevant cases correctly, but that would give a low recall. Similarly, classifying all cases as relevant would give a high recall, but give a low precision. Because of this, precision and recall give a good indication of performance together.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

The true positives (TP) determine pixels (or voxels) correctly classified as being part of the segmentation, a false positive (FP) is a pixel incorrectly classified as

being part of the segmentation, and a false negative (FN) is a pixel which should have been part of the segmentation but was not.

The Dice similarity coefficient (DSC) is a spatial overlap index and the harmonic mean of the precision and recall [14]. It is frequently used for evaluating medical segmentations. The values of Dice ranges from zero to one, where zero indicates no overlap between target and prediction, while one indicates complete overlap. For two regions, A and B, the DSC is defined as two times the intersection over the union.

$$DSC = \frac{2(A \cap B)}{A + B} = \frac{2TP}{2TP + FP + FN}$$

3.6.2 Method

All experiments were performed using four splits. Each split was made up of 75% of the training set for training and 25% for validation and final testing. Validation was performed by calculating the score for random slices from the validation set after each epoch. The training was terminated when the score on the validation set had not improved over a number of epochs.

For each experiment, the recall, precision, and Dice values were calculated. To find these statistics, a confusion matrix was calculated for every image. A confusion matrix gives an overview of which predictions are correct, while also showing what the incorrect guesses were predicted as. The rows of a confusion matrix show the actual classes, while the columns show which classes are predicted. The precision, recall, and Dice values were calculated by summing the confusion matrices of all images, from all four validation splits. This was done to get accurate statistics of the entire dataset, as predictions of each class will then be weighted by the proportion of appearance in a given image.

Furthermore, a Dice value using the split scores were calculated. This Dice value was calculated similarly to how LaLonde et al. did. For each split, the value is the mean or the median of the Dice score of each image in the corresponding validation set. As these scores are calculated differently than the overall dataset Dice value, the given average of the splits is not necessarily equal to the overall dataset Dice. This metric is not optimal from a statistical point of view, as the Dice score of an image containing a single positive class pixel will be weighed equally as an image with thousands of positive class pixels. However, it was necessary for comparing achieved results with LaLonde et al. Notice that each split score is calculated using the median in the second experiment, while the other experiments use the mean. The reason is that LaLonde et al. used the median, while we found

the mean to be more describing.

All datasets from the Medical Segmentation Decathlon (MSD) contained a test set without publicly available labels. In order to obtain official test scores, segmentations of test sets using 2.5D U-Net were submitted to the MSD. By doing so, the architecture obtained scores that could be directly compared to the winner of MSD. Dice scores obtained from MSD were calculated as the mean of the Dice scores of every image in the test set.

3.7 Environment

Most models were trained and tested using nodes from NTNU's EPIC cluster. SegCaps models were trained using Tesla P100 GPUs, while some of the 2.5D U-Net models were run on Tesla V100 GPUs. Both GPUs were from Nvidia and had 16 GB of video memory. Jobs were submitted using the Slurm² job scheduler. By using EPIC it was possible to train and test many models at the same time.

To be able to train more models in parallel, some of the 2.5D U-Net models were trained on NTNU's DGX cluster. The cluster has V100 GPUs with 32GB of video memory. Even though this gave the possibility of increasing the batch size, it was kept unchanged for all 2.5D U-Net models, in order to have comparable results.

EM-SegCaps models were trained on a local computer having a single GTX1080 GPU with 11GB of memory. This was done because of an incompatibility between the way EM-SegCaps performs matrix multiplications of large batches of matrices and the version of CUDA installed on the EPIC cluster.

²<https://slurm.schedmd.com/documentation.html>

Chapter 4

Results

This chapter presents the results of the four experiments that were set up to test different types of segmentation architectures. These architectures were 2.5D U-Net, SegCaps, Multi-SegCaps and EM-SegCaps. Each experiment trained the respective architecture on one or several medical segmentation datasets. An overview of Dice scores (in percent) achieved by a given architecture - dataset combination, based on all experiments conducted, are shown in Table 4.1 and Table 4.2. The first table shows datasets with a single target class, while the second table contains datasets with multiple target classes. The columns show a score for each target class in the respective dataset. The hippocampus dataset is represented two times as EM-SegCaps had an experiment where the originally two hippocampus classes were combined into one class. The SegCaps architecture is excluded from the second table, as it only had scores for LUNA16, heart and spleen datasets.

	LUNA16	Spleen	Heart	Hippoc.
	1	1	1	1
2.5D U-Net	-	90.0	91.3	-
SegCaps	98.2	40.4	60.4	-
Multi-SegCaps	-	50.6	67.0	-
EM-SegCaps	-	-	-	54.5

Table 4.1: An overview of Dice scores for all models, for the first part of the datasets.

	Hippoc.		Liver		Pancreas		BraTS		
	1	2	1	2	1	2	1	2	3
2.5D U-Net	84.2	82.7	91.0	68.9	69.5	18.2	65.0	26.1	34.6
Multi-SegCaps	72.4	70.5	66.1	0.2	-	-	28.5	11.3	0.1
EM-SegCaps	18.7	24.5	-	-	-	-	-	-	-

Table 4.2: An overview of Dice scores for all models, for the second part of the datasets.

4.1 2.5D U-Net for Medical Segmentation Tasks

The developed 2.5D U-Net was tested on various medical segmentation tasks. The chosen datasets aimed to segment the spleen, the heart, the anterior and posterior hippocampus, the liver and liver tumour, the pancreas and pancreas tumour, as well as three classes of brain tumour. These were selected for testing the architecture on medical segmentation problems that are challenging in different ways.

Each of the datasets had a test set without publicly known labels. The segmentation masks for these test sets were submitted to the Medical Segmentation Decathlon. The Dice scores obtained are shown in Table 4.3. The figure also shows the scores of the nnU-Net, which won the challenge in 2018. 2.5D U-Net does not manage to achieve higher scores than nnU-Net on any of the tasks. However, some scores are only a few percentage points lower. The rest of this section shows the scores calculated from the validation set when using four splits.

	Spleen	Heart	Hippoc.	Liver	Pancreas	BraTS
	1	1	1 2	1 2	1 2	1 2 3
nnU-Net	96	93	90 89	95 74	80 52	68 48 68
2.5D U-Net	85	72	87 85	90 54	71 26	53 28 50

Table 4.3: Dice scores from the test sets, obtained from Medical Segmentation Decathlon.

4.1.1 Spleen

Table 4.4 shows the performance of the 2.5D U-Net when trained on the spleen dataset. The performance of the model using one slice as input is slightly lower compared to the model using five consecutive input slices for predicting the middle slice. For both experiments the recall is higher than the precision, indicating

that the prediction had few false negatives, but some of the background pixels were incorrectly predicted as the spleen class. The performance on the splits is shown in Table 4.5 for both the single slice and the five slice model. The table of split scores shows a 10 percentage point difference between the best and the worst performing split. An example segmentation using five consecutive input slices can be seen in Figure 4.1. A comparison of predictions and the ground truth label on a failure case is shown in Figure 4.2. The figure shows that this particular volume suffered from severe under-segmentation, as well as slight over-segmentation of irrelevant background. An activation from the output layer in 2.5D U-Net for a spleen volume is visualized in Figure A.1.

The average Dice scores of the two tables are not equal. The first table shows the average of summing the confusion matrices of all volumes, from all four validation splits. The second table’s Dice score is the average of the four splits scores, each being the mean of the Dice score of each volume in the corresponding validation set.

	Recall	Precision	Dice
Spleen (1 slice)	93.234	86.097	89.524
Spleen (5 slice)	93.185	87.111	90.046

Table 4.4: Percentages of metrics totalled over the spleen dataset.

	Split-0	Split-1	Split-2	Split-3	Average
Spleen (1 slice)	92.468	88.984	82.529	84.665	87.161
Spleen (5 slice)	92.373	89.256	82.861	87.773	88.066

Table 4.5: Percentages of correct classification (Dice score) on different splits of the spleen dataset. Numbers reported in mean.

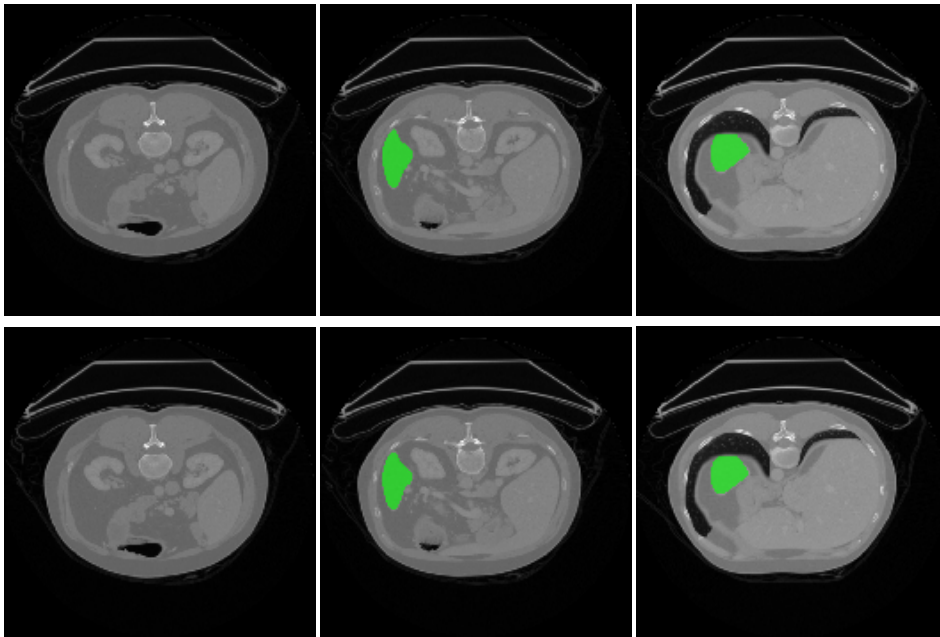


Figure 4.1: Slices from an example volume from spleen segmentation using 2.5D U-Net with 5 slices as input. The upper row shows predictions, while the second row shows ground truth. The Dice score of this particular volume was 95.772%.

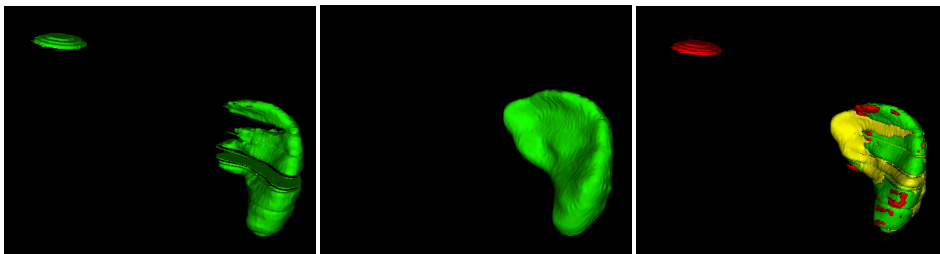


Figure 4.2: A 3D visualization of spleen segmentation using 2.5D U-Net. From left to right: prediction, ground truth and both combined. In the third image green is true positives, red is false positives and yellow is false negatives. The volume achieved a Dice score of 74.824%

4.1.2 Heart

The performance obtained on the heart dataset is shown in Table 4.6. The scores for all the splits are shown in Table 4.7. Despite being a small dataset of only 20 volumes, the model performed reasonably well on all splits. Figure 4.3 shows predictions made by the model. It shows that the model is able to capture small parts of the left atrium, as well as larger parts.

	Recall	Precision	Dice
Left atrium	90.690	92.114	91.396

Table 4.6: Percentages of metrics totalled over the heart dataset.

	Split-0	Split-1	Split-2	Split-3	Average
Left atrium	90.477	92.130	90.763	91.883	91.313

Table 4.7: Percentages of correct classification (Dice score) on different splits of the heart dataset. Numbers reported in mean.

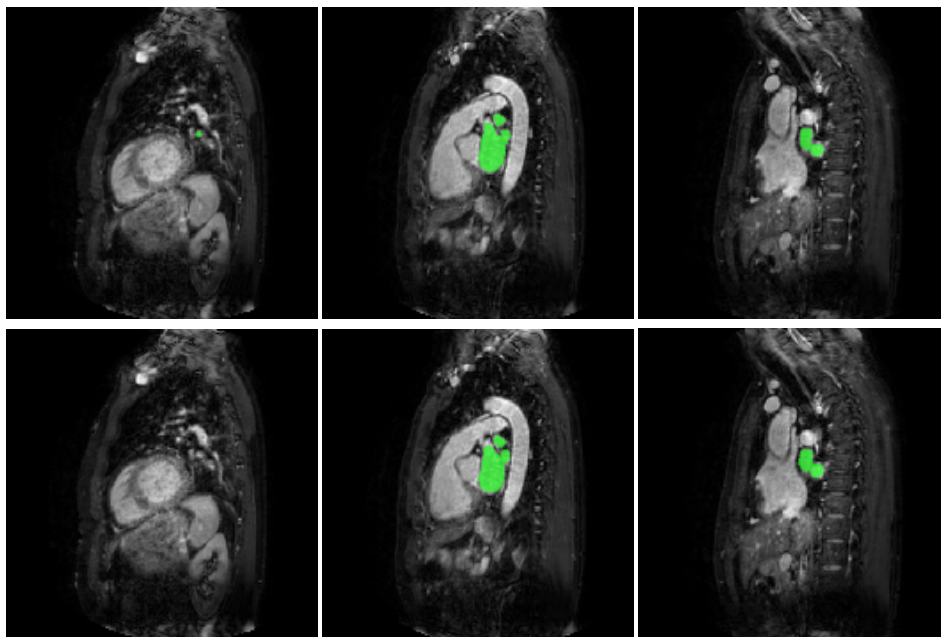


Figure 4.3: Slices from an example volume from segmentation of the left atrium. The upper row shows predictions, while the second row shows ground truth. The score of this particular volume was 92.711%.

4.1.3 Hippocampus

The scores achieved after training the 2.5D U-Net model on the hippocampus dataset are shown in Table 4.8. The model was able to achieve similar results

for both anterior and posterior hippocampus classes. Further, as shown in Table 4.9, the differences in performance for the splits were small. When compared to state of the art results by nnU-Net (Table 4.3), 2.5D U-Net achieved only 3% and 4% lower scores for anterior and posterior classes, respectively. An example of segmentation can be seen in Figure 4.5. The middle slice shows how the model incorrectly predicts some of the posterior class (red) as being the anterior hippocampus (green). A 3D visualization of a segmentation is shown in Figure 4.4.

	Recall	Precision	Dice
Anterior	84.546	85.818	85.177
Posterior	81.433	86.541	83.909

Table 4.8: Percentages of metrics totalled over the hippocampus dataset.

	Split-0	Split-1	Split-2	Split-3	Average
Anterior	84.270	84.939	84.986	82.797	84.248
Posterior	83.374	83.477	82.466	81.601	82.730

Table 4.9: Percentages of correct classification (Dice score) on different splits of the hippocampus dataset. Numbers reported in mean.

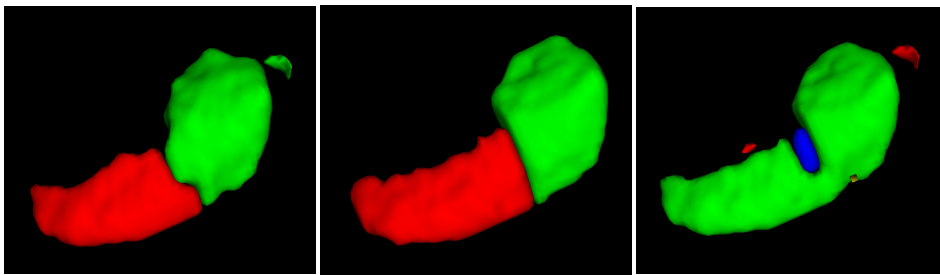


Figure 4.4: A 3D visualization of hippocampus segmentation using 2.5D U-Net. From left to right: prediction, ground truth and both combined. In the third image green is true positives, red is false positives, yellow is false negatives and blue is anterior predicted as posterior, or vice versa.

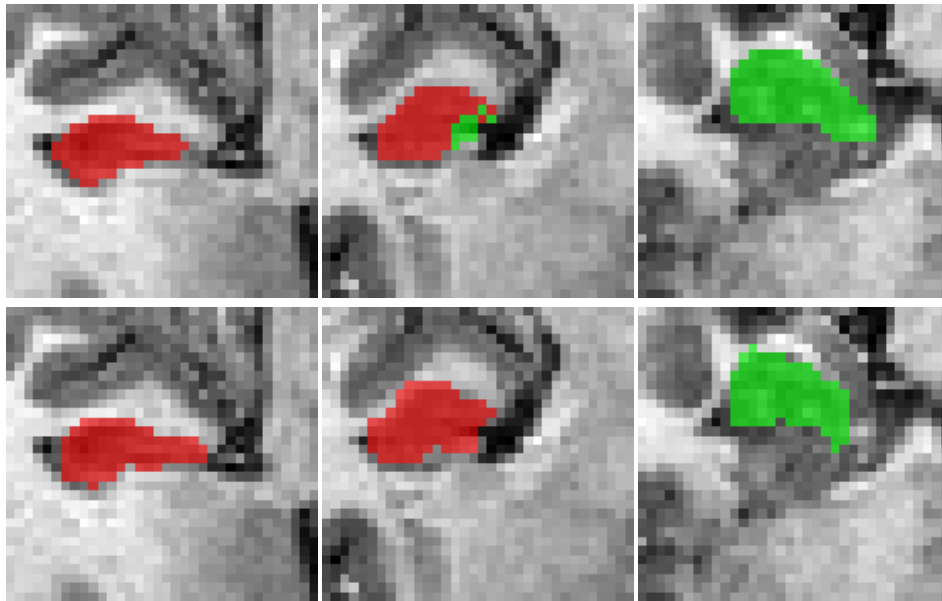


Figure 4.5: Slices from an example volume from hippocampus segmentation. The upper row shows predictions, while the second row shows ground truth. The score of this particular volume was 85.138% for anterior hippocampus (green) and 84.458% for posterior hippocampus (red).

4.1.4 Liver

The architecture was able to learn the liver segmentation well, as indicated by Table 4.10. However, Figure 4.7 shows that the scores for the segmentation of liver class were steady while the scores for liver tumour were highly variable. This is also reflected in Table 4.11, which shows the same trend for the splits. 2.5D U-Net achieved 5% and 20% lower scores than nnU-Net for liver and liver tumour, respectively (Table 4.3). A representative volume for the general performance is shown in Figure 4.6. The volume shows that the model predicts the tumour in the correct sport, but misses some pixels.

	Recall	Precision	Dice
Liver	90.651	91.321	90.985
Tumour	62.755	76.445	68.927

Table 4.10: Percentages of metrics totalled over the liver dataset.

	Split-0	Split-1	Split-2	Split-3	Average
Liver	92.061	91.581	88.461	90.070	90.543
Tumour	34.285	51.836	37.525	46.003	42.412

Table 4.11: Percentages of correct classification (Dice score) on different splits of the liver dataset. Numbers reported in mean.

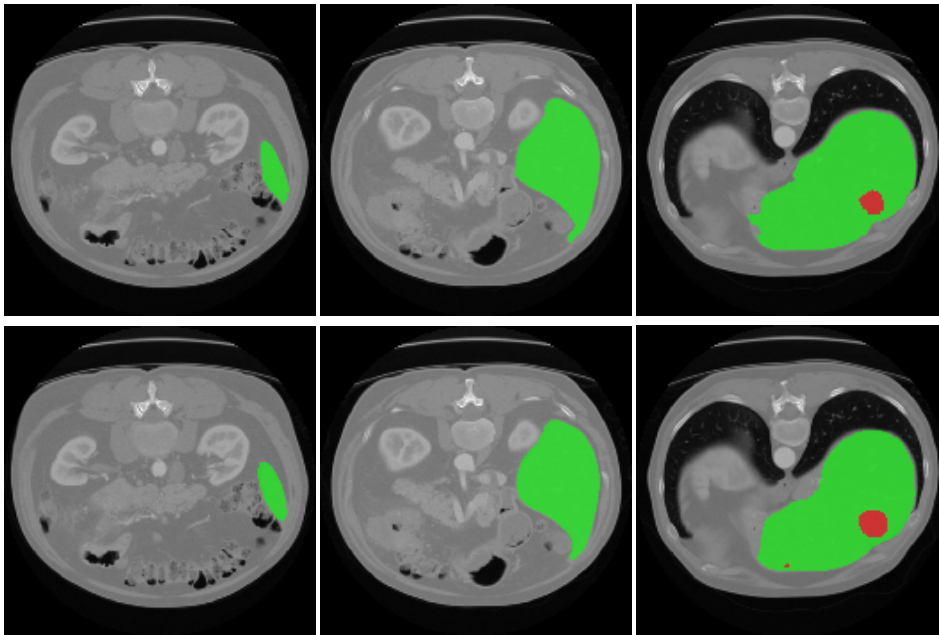


Figure 4.6: Slices from an example volume from liver segmentation. The upper row shows predictions, while the second row shows ground truth. The Dice score of this particular volume was 93.652% for liver (green) and 64.324% for liver tumour (red).

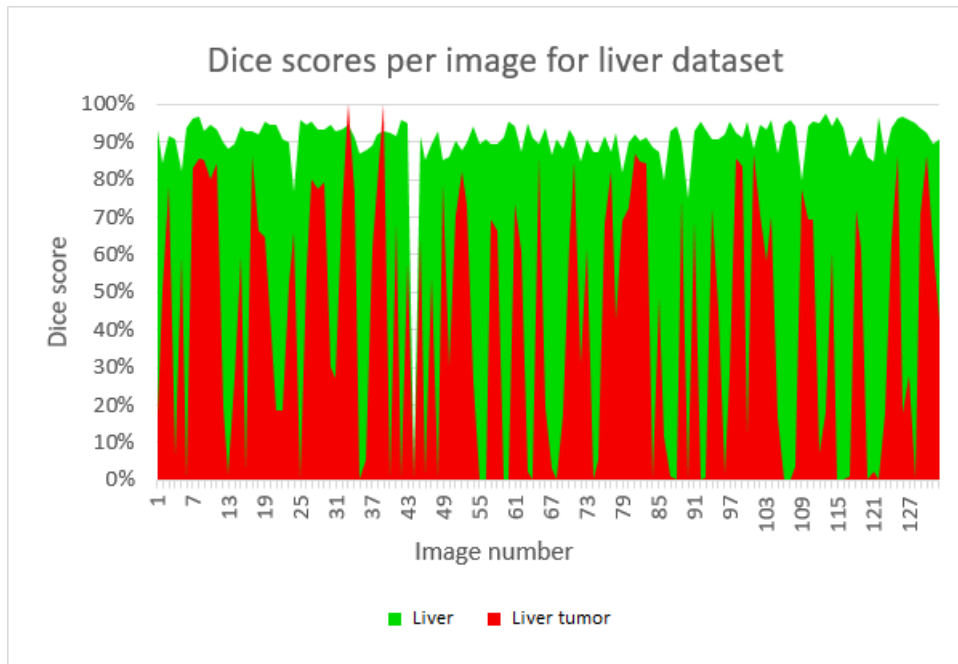


Figure 4.7: Dice scores per volume for the liver dataset using 2.5D U-Net.

4.1.5 Pancreas

Results of applying the 2.5D U-Net model to the pancreas dataset are presented in Table 4.12. Despite containing more challenging structures than the previously presented datasets, the performance on the pancreas test set was only 9% lower than for nnU-Net for the pancreas class (Table 4.3). However, the model does not manage to learn pancreas tumour very well. The low precision for tumour indicates that the model misclassifies background and pancreas as being of the tumour class, as shown in Figure 4.8. The low recall indicates that pancreas tumour is often incorrectly predicted as pancreas or background. Even though each split contains about 70 volumes, the performance of the splits is varying. The scores for each split are shown in Table 4.13.

	Recall	Precision	Dice
Pancreas	68.840	70.094	69.461
Tumour	12.746	31.663	18.175

Table 4.12: Percentages of metrics totalled over the pancreas dataset.

	Split-0	Split-1	Split-2	Split-3	Average
Pancreas	67.744	69.471	74.087	63.134	68.609
Tumour	23.654	20.506	22.600	9.836	19.149

Table 4.13: Percentages of correct classification (Dice score) on different splits of the pancreas dataset. Numbers reported in mean.

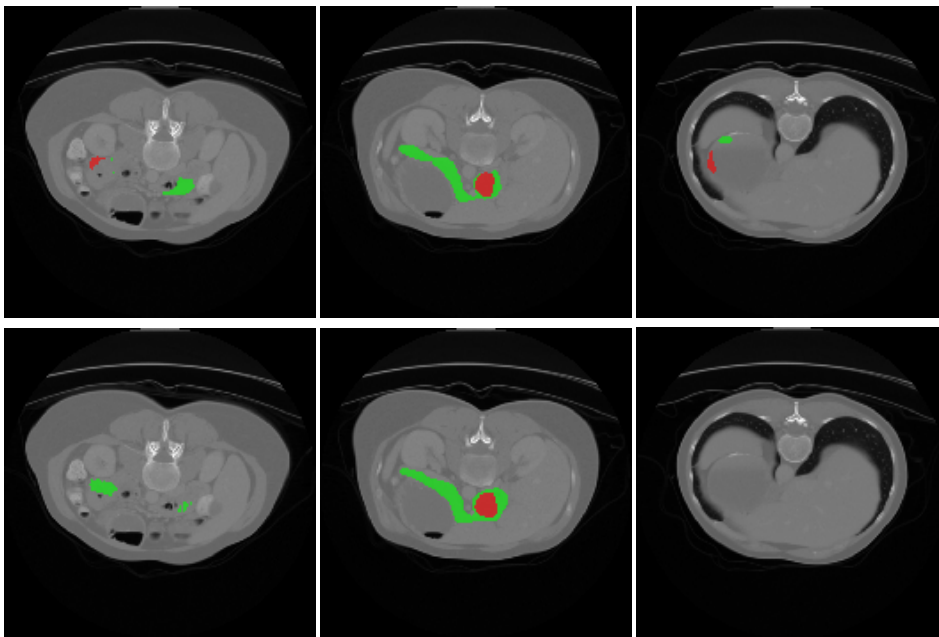


Figure 4.8: Slices from an example volume from pancreas segmentation. The upper row shows predictions, while the second row shows ground truth. The Dice scores of this particular volume was 50.691% for pancreas (green) and 53.131% for pancreas tumour (red).

4.1.6 BraTS

Table 4.14 shows the results obtained when fitting the 2.5D U-Net to the BraTS dataset. The scores for each split are shown in Table 4.15. The first two splits achieved higher scores on all classes than the two last splits. Figure 4.9 shows an example of one of the better predictions made by the model. It shows that the model mostly captures all of the tumour, but confuses the classes.

	Recall	Precision	Dice
Edema	67.263	62.861	64.988
Non enh.	33.458	21.427	26.124
Enh.	62.211	23.977	34.613

Table 4.14: Percentages of metrics totalled over the BraTS dataset.

	Split-0	Split-1	Split-2	Split-3	Average
Edema	68.806	74.531	50.010	43.037	59.096
Non enh.	30.522	37.619	22.791	19.292	27.556
Enh.	64.926	63.825	41.414	41.069	52.809

Table 4.15: Percentages of correct classification (Dice score) on different splits of the BraTS dataset. Numbers reported in mean.

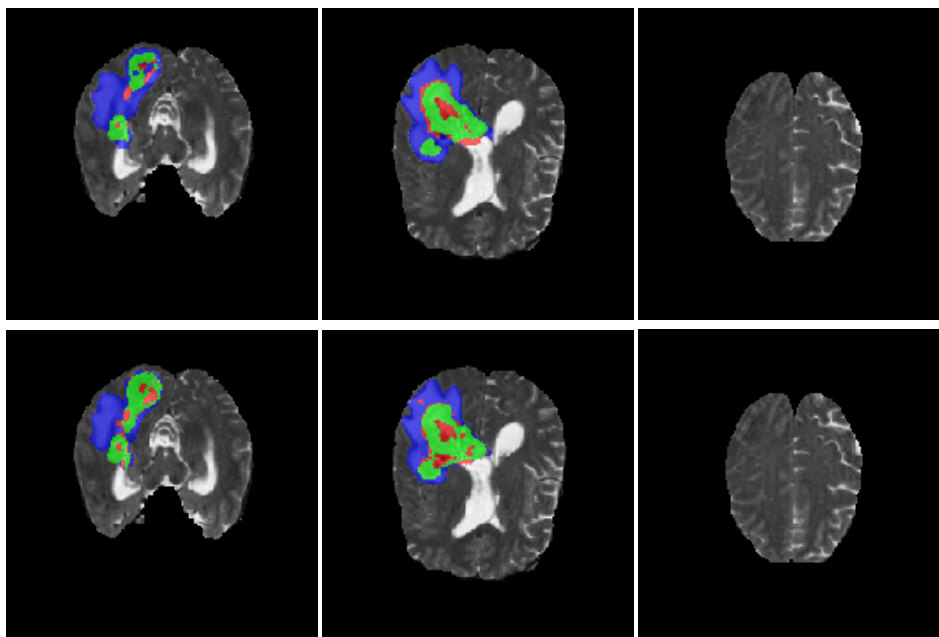


Figure 4.9: Slices from an example volume from BraTS segmentation. The upper row shows predictions, while the second row shows ground truth. The Dice scores of this particular volume were 81.853% for edema (blue), 63.961% for non-enhancing tumour (red) and 85.966% for enhancing tumour (green).

4.2 Reproduce Results from *Capsules for Object Segmentation*

The SegCaps architecture presented in the paper *Capsules for Object Segmentation* was trained on the LUNA16 dataset as well as the spleen and the heart datasets. LaLonde et al. had applied the architecture to the LUNA16 dataset, and the scores they reported were compared to the achieved scores. The spleen and the heart datasets were used to investigate how applicable the same model was to similar problems.

The number of epochs until the models early stopped, as well as training time for the three datasets are shown in Table A.1. Training of the LUNA16 dataset took about 2-4 days, the spleen dataset spent about 3-5 days and the heart dataset spent 1-2 days.

4.2.1 LUNA16

The first row of Table 4.17 shows our scores for the LUNA16 dataset. The score reported by LaLonde et al. in *Capsules for Object Segmentation* is shown in the second row. They did not report which volumes their four splits were made up of, so the scores for each split is not directly comparable. However, all split scores of LaLonde et al. were higher than the ones achieved in this experiment. This makes our median Dice score slightly lower than the one reported in the literature. Examples of segmentations made are shown in Figure 4.10 and Figure 4.11. The latter figure shows a volume where the label appears to be wrong, while the prediction is fine, revealing a weakness of the LUNA16 segmentation data. This is reflected in Figure 4.12 which shows that the scores for a few volumes were significantly lower.

An overview of total precision, recall and Dice score is shown in Figure 4.16. While calculating these scores, a mistake in the source code by LaLonde et al. was discovered and fixed. The test code did not use only the lung target class for evaluation, but rather the original four segmentation classes from LUNA16. Fixing this issue resulted in slightly different scores for the splits, given in Table A.3. However, as the authors had used the test code with this mistake, we used our average split score, calculated in the same way as theirs, for comparing the results.

The average Dice scores of Figure 4.16 and Figure 4.17 are not equal. The first table shows the average of summing the confusion matrices of all volumes, from all four validation splits. The second table's Dice score is the average of the four

splits scores, each being the median of the Dice score of each volume in the corresponding validation set.

	Recall	Precision	Dice
Lungs (Our results)	97.628	97.129	97.378

Table 4.16: Percentages of metrics totalled over the LUNA16 dataset.

	Split-0	Split-1	Split-2	Split-3	Average
Lungs (Our results)	98.467	98.189	98.072	98.238	98.242
Lungs (LaLonde et al.)	98.499	98.523	98.455	98.474	98.479

Table 4.17: Percentages of correct classification (Dice score) on different splits of the LUNA16 dataset. Numbers reported in median.

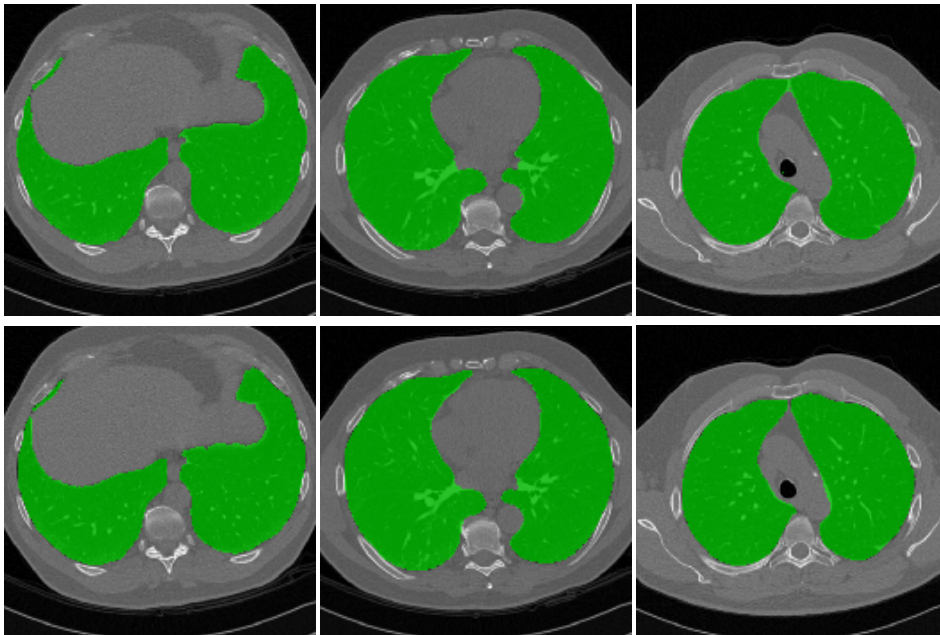


Figure 4.10: Example slices from LUNA16 segmentation using SegCaps. The top row shows predictions, while the bottom row shows ground truth. The score of this particular volume was 97.822 %.

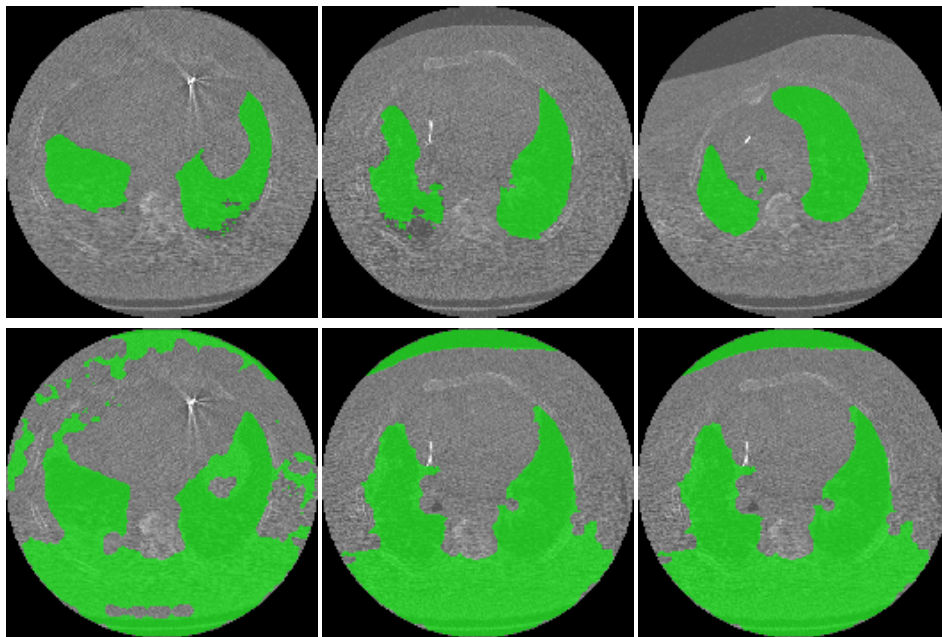


Figure 4.11: Example slices from LUNA16 segmentation using SegCaps of a volume with an incorrectly labelled ground truth. The upper row shows predictions, while the second row shows ground truth. The score of this particular volume was 30.117 %.

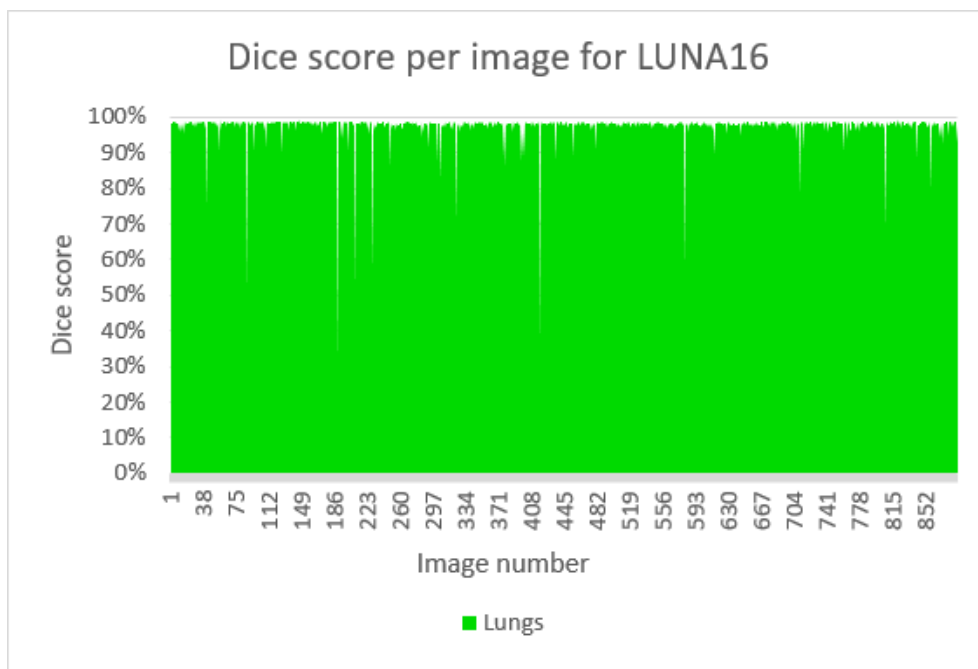


Figure 4.12: Dice scores per volume for the LUNA16 dataset using SegCaps.

4.2.2 Spleen

The SegCaps architecture was applied to the spleen dataset, achieving the scores shown in Table 4.18. The performance is significantly lower than for the LUNA16 dataset, and the Dice score is below half of the one obtained by the U-Net model on the same dataset. Table 4.19 displays that the variation of performance achieved by the different splits is high. The table gives each split score in both medians, similarly to how LaLonde et al. calculated their scores for LUNA16, and in mean, as the results from the other experiments were reported. Figure 4.13 shows predictions made by the model. The first slice shows how the model wrongly predicts another body part as being the spleen.

	Recall	Precision	Dice
Spleen	97.214	25.494	40.394

Table 4.18: Percentages of metrics totalled over the spleen dataset.

	Split-0	Split-1	Split-2	Split-3	Average
Spleen (Median)	59.838	56.523	35.911	20.967*	43.310
Spleen (Mean)	61.320	47.233	41.832	23.354*	43.438

Table 4.19: Percentages of correct classification (Dice score) on different splits of the spleen segmentation dataset. (*) Result from split 3 obtained from second training of same split. Results from first training were 7.515% (median) and 7.979% (mean).

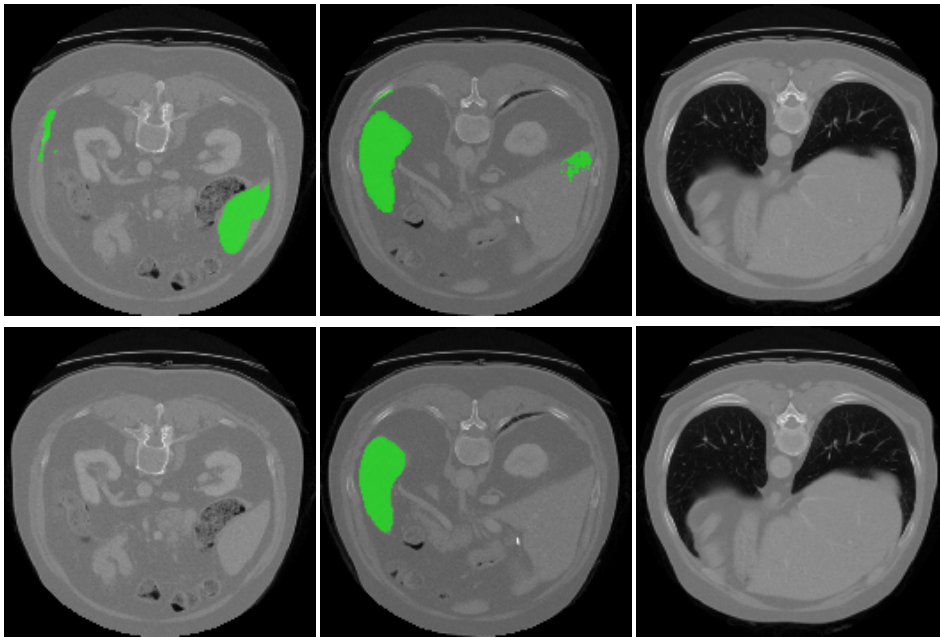


Figure 4.13: Example slices from spleen segmentation using SegCaps. The upper row shows predictions, while the second row shows ground truth. The score of this particular volume was 60.638 %.

4.2.3 Heart

SegCaps was also trained to segment left atrium, obtaining the results indicated by Table 4.20. Compared to the U-Net model the average Dice score for this model was about 30% lower. As for the spleen model, the model shows the ability to achieve high recall, but the precision is poor. Figure 4.14 shows how the model captures the heart well, but wrongly segments some of the background as the target class. The scores for each split is given in Table 4.21. The difference in score from the lowest to the highest performing model was about 20%. The table gives each split score in both median, similarly to how LaLonde et al. calculated their scores for LUNA16, and in mean, as the results from the other experiments were reported.

	Recall	Precision	Dice
Left atrium	96.348	43.962	60.376

Table 4.20: Percentages of metrics totalled over the heart dataset.

	Split-0	Split-1	Split-2	Split-3	Average
Left atrium (Median)	70.550	57.292	68.366	50.922	61.783
Left atrium (Mean)	68.999	61.751	68.818	48.804	62.091

Table 4.21: Percentages of correct classification (Dice score) on different splits of the heart segmentation dataset.

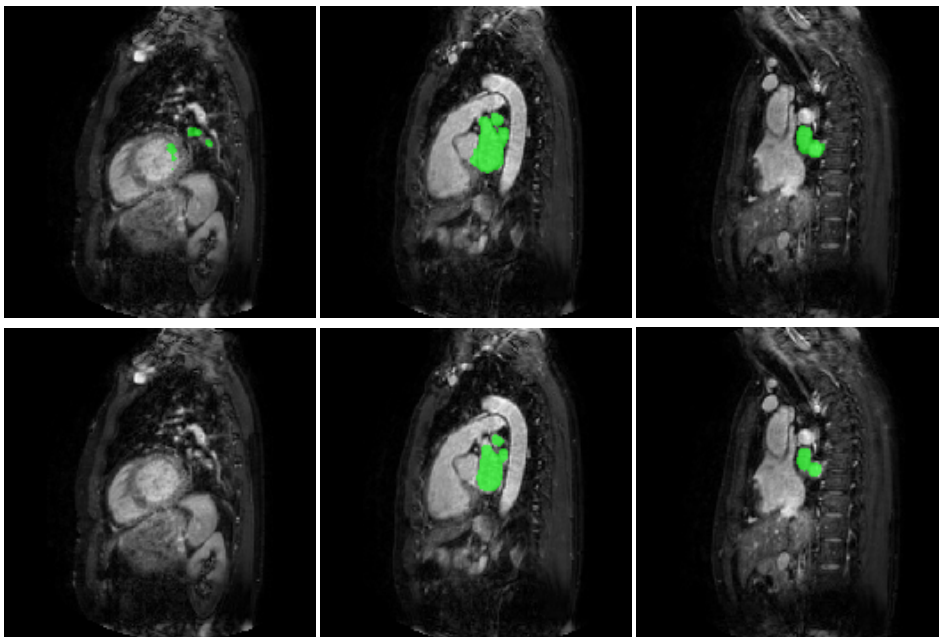


Figure 4.14: Example slices from heart segmentation using SegCaps. The upper row shows predictions, while the second row shows ground truth. The Dice score of this particular volume was 67.958 %.

4.3 Multi-class Segmentation using SegCaps

The implemented Multi-SegCaps architecture was applied to several medical segmentation datasets. These datasets were spleen, heart, hippocampus, liver, and BraTS. The goal was to see if the architecture performed as well as the regular binary SegCaps and to investigate if it was able to learn to segment several classes.

4.3.1 Spleen

By segmenting the spleen dataset it was possible to compare the performance to the original SegCaps architecture. The spleen dataset was also used for experimenting with loss functions. The loss functions tested were Dice loss and weighted cross entropy loss.

Table 4.22 and Table A.4 show that the Dice scores were 0 when using Dice loss for Multi-SegCaps. The loss graph in Figure A.2 shows that the validation score did not improve after epoch 28, even when training for 120 epochs. An example of result and activations can be seen in Figure A.3 and Figure A.4. This shows that when using Dice loss the predictions were solely background.

	Recall	Precision	Dice
Spleen	0	N/A*	0

Table 4.22: Percentages of metrics totalled over the spleen dataset when using Dice loss. (*) The value is not defined because there were no true positives or false positives.

Table 4.23 shows the performance on the spleen dataset when using weighted cross-entropy as the loss function. The score on the different splits are shown in Figure 4.24. The variation in score between the splits is high. Further, it was discovered that training one split several times might yield different results. When retraining the lowest scoring split (split 2), in the exact same way, it achieved around 40% higher Dice score. The table shows that the overall performance is higher than for the binary SegCaps model. An example of predictions can be seen in Figure 4.15.

The Dice scores of Table 4.23 and Table 4.24 are not equal. The first table shows the average of summing the confusion matrices of all volumes, from all four validation splits. The second table's Dice score is the average of the four splits scores, each being the mean of the Dice score of each volume in the corresponding validation set.

	Recall	Precision	Dice
Spleen	78.104	37.469	50.643

Table 4.23: Percentages of metrics totalled over the spleen dataset when using weighted cross-entropy loss.

	Split-0	Split-1	Split-2	Split-3	Average
Spleen	41.637	46.512	64.374*	59.840	53.341

Table 4.24: Percentages of correct classification (Dice score) on different splits of the spleen dataset when using weighted cross-entropy. Numbers reported in mean. (*) Second training of same split, result from first training was 21.552%

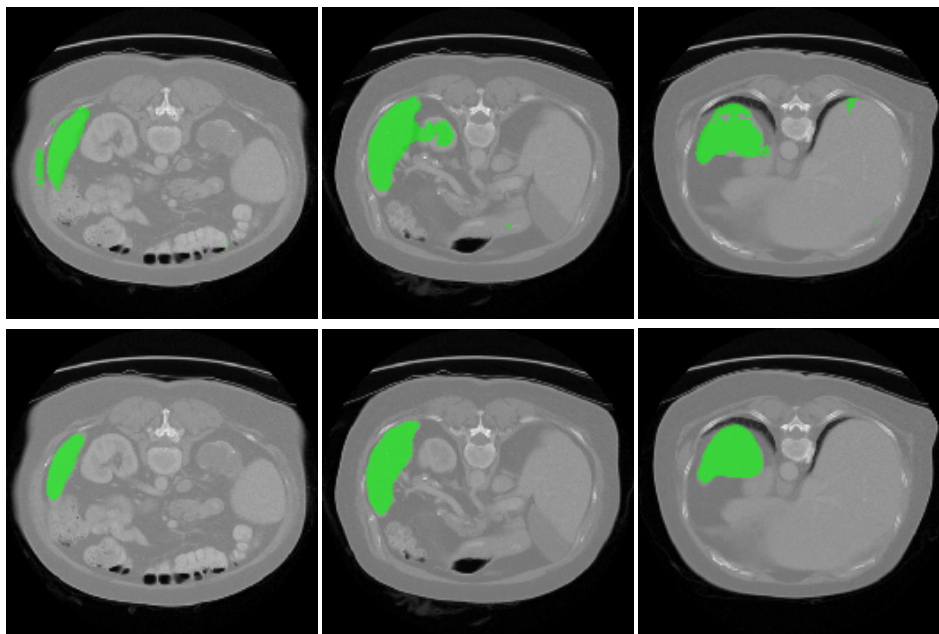


Figure 4.15: Slices from an example volume from spleen segmentation. The upper row shows predictions, while the second row shows ground truth. The score of this particular volume was 80.307%.

4.3.2 Heart

Training Multi-SegCaps on the heart dataset gave higher scores than those achieved by the original SegCaps architecture. The scores are shown in Table 4.25. Similarly to the previous results, a variation in score between the splits are shown. Table 4.26 shows the Dice scores for each split. An example of segmentation is shown in Figure 4.16. As indicated by the low precision, the model segments some false positives.

	Recall	Precision	Dice
Left atrium	86.892	54.467	66.960

Table 4.25: Percentages of metrics totalled over the heart dataset.

	Split-0	Split-1	Split-2	Split-3	Average
Left atrium	68.000	60.851	77.237	66.763	68.213

Table 4.26: Percentages of correct classification (Dice score) on different splits of the heart dataset. Numbers reported in mean.

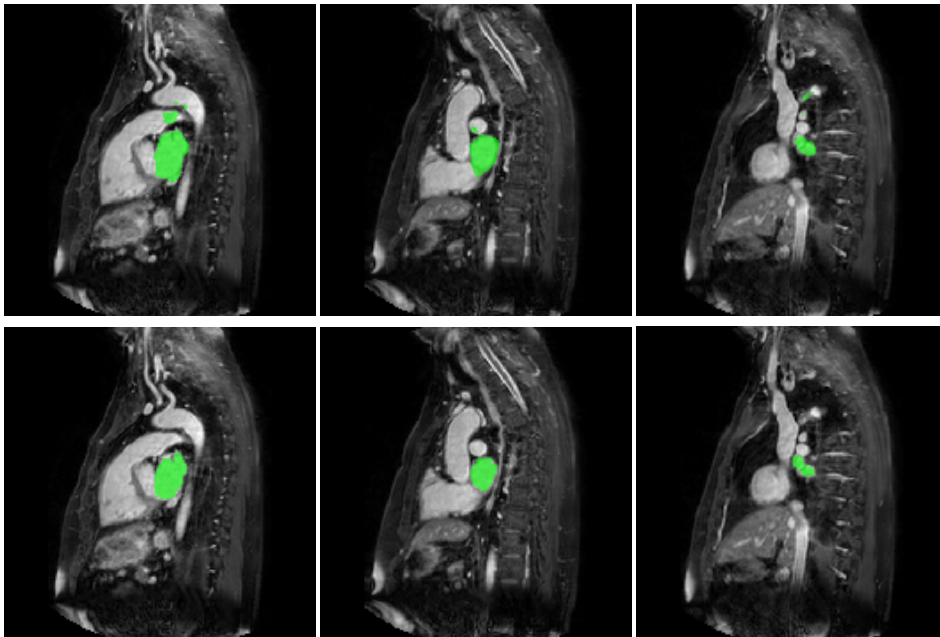


Figure 4.16: Slices from an example volume from heart segmentation. The upper row shows predictions, while the second row shows ground truth. The Dice score of this particular volume was 76.778%.

4.3.3 Hippocampus

The performance on the hippocampus dataset is shown in Table 4.27. The Dice scores for anterior and posterior hippocampus are fairly good, showing that the architecture was able to learn a multi-class problem. Table 4.28 shows the results on each split. An example of segmentation is shown in Figure 4.17. The figure shows that there is some confusion between the two classes. A 3D visualization of a segmented volume is shown in Figure 4.18. The visualization shows that the model is able to capture the main structures of the hippocampus, but struggles to determine the class membership of the pixels at the borders. It also wrongly predicts some of the background as the anterior hippocampus.

	Recall	Precision	Dice
Anterior	80.764	65.645	72.424
Posterior	84.455	60.493	70.494

Table 4.27: Percentages of metrics totalled over the hippocampus dataset.

	Split-0	Split-1	Split-2	Split-3	Average
Anterior	69.835	70.378	73.584	71.997	71.449
Posterior	69.716	67.812	68.127	75.183	70.210

Table 4.28: Percentages of correct classification (Dice score) on different splits of the hippocampus dataset. Numbers reported in mean.

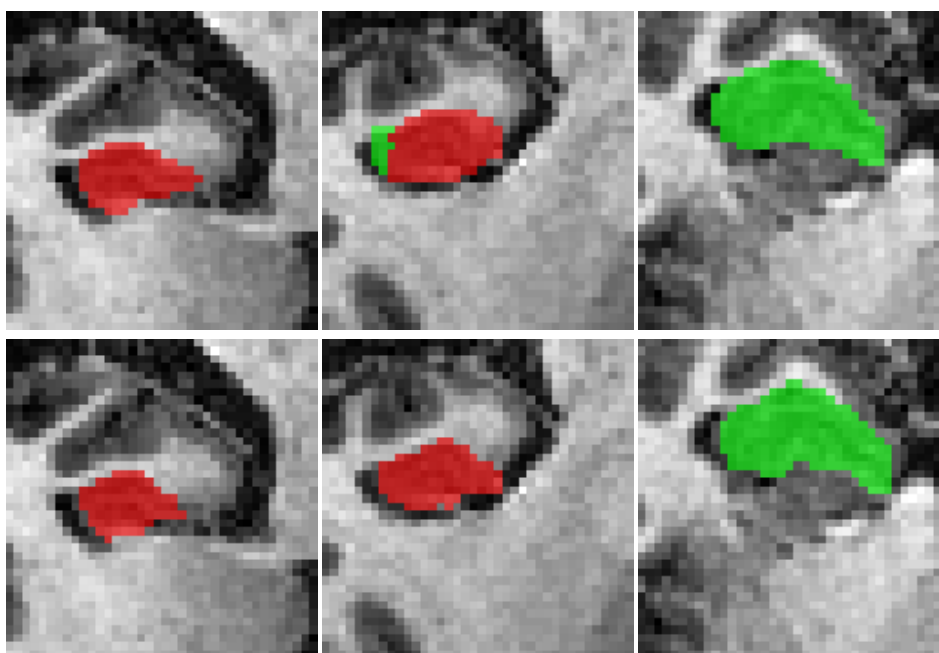


Figure 4.17: Slices from an example volume from hippocampus segmentation. The upper row shows predictions, while the second row shows ground truth. The Dice score of this particular volume was 76.549% for anterior hippocampus (green) and 79.122% for the posterior hippocampus (red).

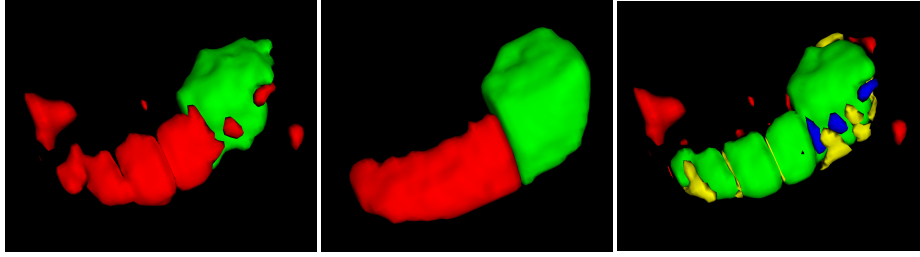


Figure 4.18: A 3D visualization of hippocampus segmentation using Multi-SegCaps. From left to right: prediction, ground truth and both combined. In the third image green is true positives, red is false positives, yellow is false negatives and blue is anterior predicted as posterior, or vice versa.

4.3.4 Liver

The performance of Multi-SegCaps on the liver dataset is shown in Table 4.29. The score for the liver is reasonably good, but the model did not seem to learn the liver tumour class. Figure 4.19 shows an example of liver and tumour segmentation. In the rightmost slice, we see one of the few examples where the model started to learn how to segment tumour structures. The example achieved above average performance, and is an example where parts of a tumour structure are correctly segmented. The scores for each split is given in Table 4.30. It shows that the different splits have quite different scores. An illustration of the scores (Figure 4.20) shows that the liver class is learned pretty well, but not as good as when using the 2.5D U-Net model (Figure 4.7). The volumes with a perfect Dice score for liver tumour did not contain any tumour.

	Recall	Precision	Dice
Liver	89.712	52.335	66.106
Tumour	0.098	3.210	0.191

Table 4.29: Percentages of metrics totalled over the liver dataset.

	Split-0	Split-1	Split-2	Split-3	Average
Liver	67.440	73.790	69.897	56.419	66.887
Tumour	15.152	9.091	1.146	9.375	8.691

Table 4.30: Percentages of correct classification (Dice score) on different splits of the liver dataset. Numbers reported in mean.

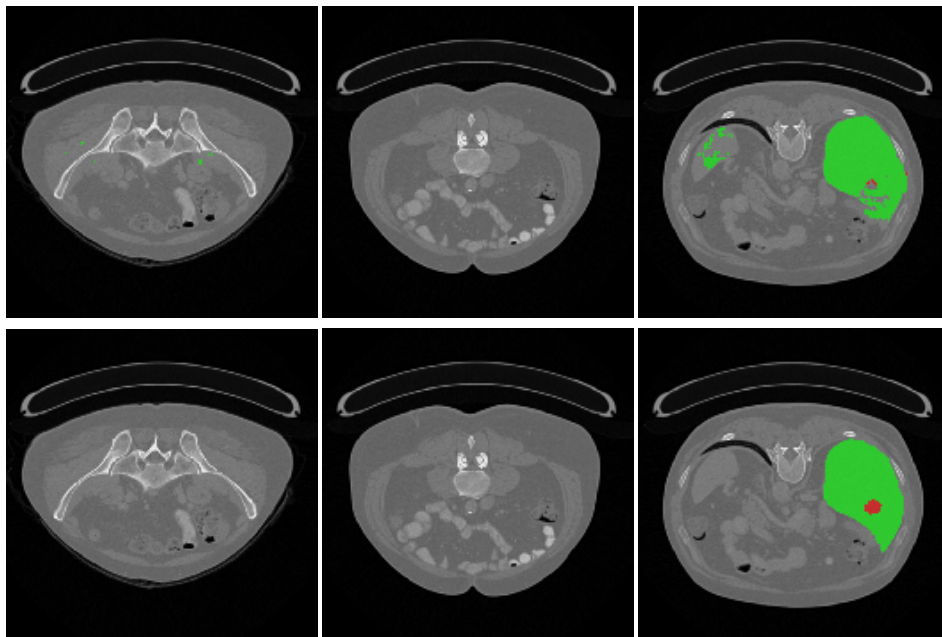


Figure 4.19: Slices from an example volume from liver segmentation. The upper row shows predictions, while the second row shows ground truth. The Dice score of this particular volume was 83.010% for liver (green) and 8.968% for liver tumour (red).

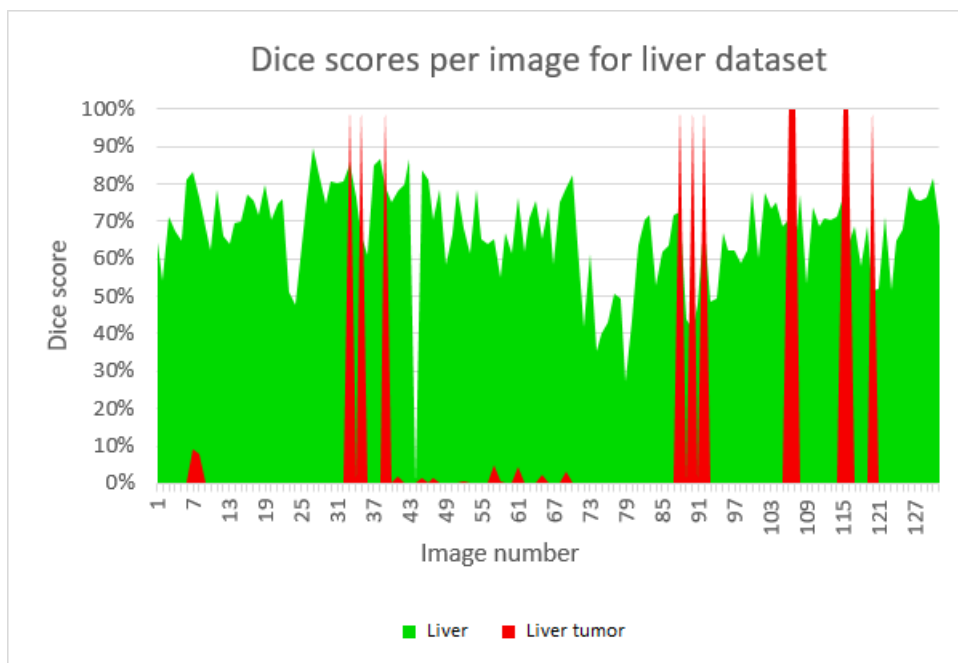


Figure 4.20: Dice scores per volume for liver dataset using Multi-SegCaps.

4.3.5 BraTS

The results on the BraTS dataset are presented in Table 4.31. The scores for enhancing tumour are poor, while the model seems to have learned some of the edema and the non-enhancing tumour classes. Table 4.32 shows the scores for each split. As shown in Figure 4.21, the model seems to learn the location of the tumour but predicts most of it as being edema.

	Recall	Precision	Dice
Edema	66.236	18.195	28.548
Non enh.	6.712	34.937	11.261
Enh.	0.027	59.238	0.055

Table 4.31: Percentages of metrics totalled over the BraTS dataset.

	Split-0	Split-1	Split-2	Split-3	Average
Edema	34.032	32.908	35.227	28.344	32.628
Non enh.	4.031	9.955	10.416	4.716	7.280
Enh.	0.000	0.191	0.000	0.000	0.048

Table 4.32: Percentages of correct classification (Dice score) on different splits of the BraTS dataset. Numbers reported in mean.

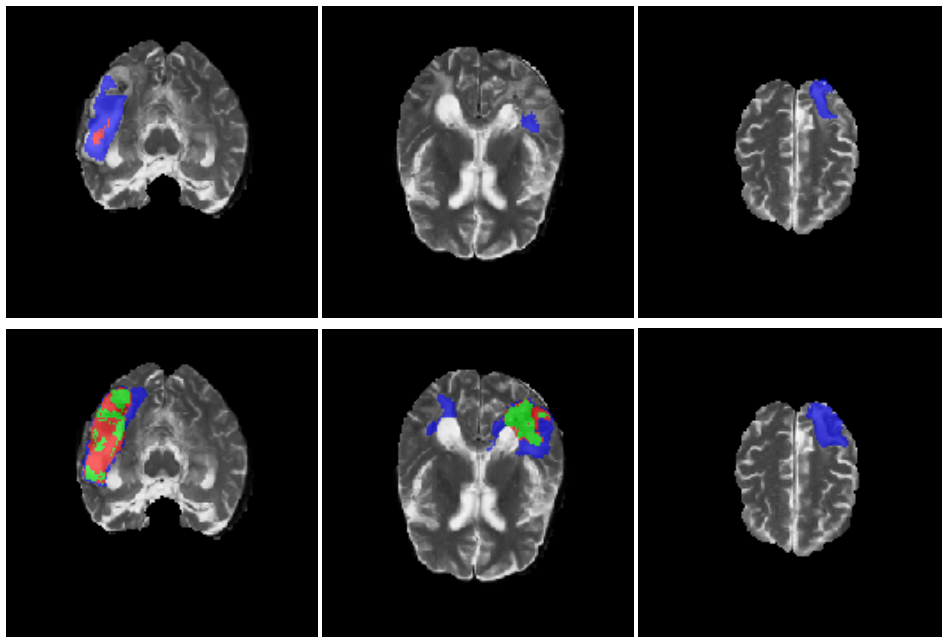


Figure 4.21: Slices from an example volume from BraTS segmentation. The upper row shows predictions, while the second row shows ground truth. The Dice score of this particular volume was 32.295% for the edema class (blue), 6.785% for the non-enhancing class (red) and 0.000% for the enhancing class (green).

4.4 SegCaps with EM-routing

The SegCaps architecture using EM-routing was tested by both using the hippocampus dataset as a binary problem and as a multi-class problem. The architecture was equal in the two cases, the only difference was that the first model used two output classes, where the background is one class, and the second model used three.

4.4.1 Binary Hippocampus

Table 4.34 shows that EM-Segcaps achieves Dice scores in the range of about 50-60% for all four splits when performing binary segmentation of volumes from the hippocampus dataset. The overall Dice score of the entire dataset was 54.500%. Figure 4.22 and Table 4.33 shows that the model correctly classifies most pixels belonging to the hippocampus class, but also incorrectly classifies some irrelevant structures as being the hippocampus.

	Recall	Precision	Dice
Hippocampus	71.264	44.122	54.500

Table 4.33: Percentages of metrics totalled over the hippocampus dataset.

	Split-0	Split-1	Split-2	Split-3	Average
Hippocampus	61.564	52.735	54.021	51.090	54.853

Table 4.34: Percentages of correct classification (Dice score) on different splits of the hippocampus dataset when considering posterior and anterior hippocampus as a single class. Numbers reported in mean.

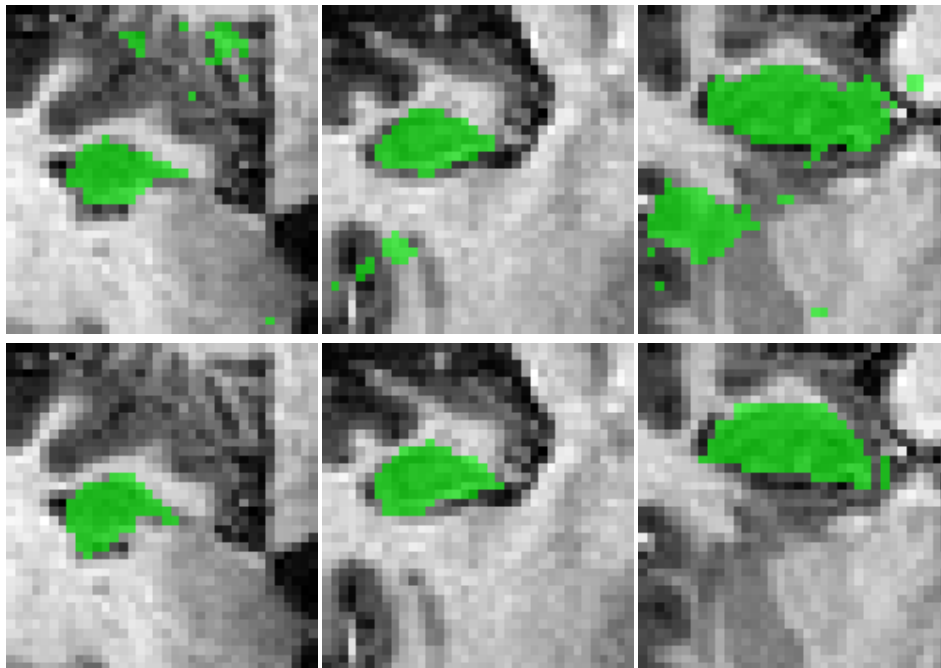


Figure 4.22: Slices from an example volume from hippocampus segmentation. The upper row shows predictions, while the second row shows ground truth. The Dice score of this particular volume was 58.279%.

4.4.2 Multi-class hippocampus

Table 4.36 shows that EM-SegCaps achieves mixed results when performing multi-class segmentation of volumes from the hippocampus dataset. The model simply predicts everything as background for two splits, giving Dice scores of zero. Split 0 and split 3 are capable of segmenting some structures. Split 0 predicts every pixel to either background or posterior hippocampus, ignoring the anterior hippocampus (Figure 4.23). Split 3 does the complete opposite and predicts some structures as anterior while ignoring the posterior class altogether (Figure 4.24). Table 4.35 provides the precision, recall, and Dice for the entire hippocampus dataset. These results are significantly worse than the ones obtained using the multi-class SegCaps and the 2.5D U-Net model.

	Recall	Precision	Dice
Anterior	17.508	20.006	18.674
Posterior	19.008	34.548	24.523

Table 4.35: Percentages of metrics totalled over the hippocampus dataset.

	Split-0	Split-1	Split-2	Split-3	Average
Anterior	0	0	0	31.470	7.868
Posterior	46.971	0	0	0	11.743

Table 4.36: Percentages of correct classification (Dice score) on different splits of the hippocampus dataset. Numbers reported in mean.

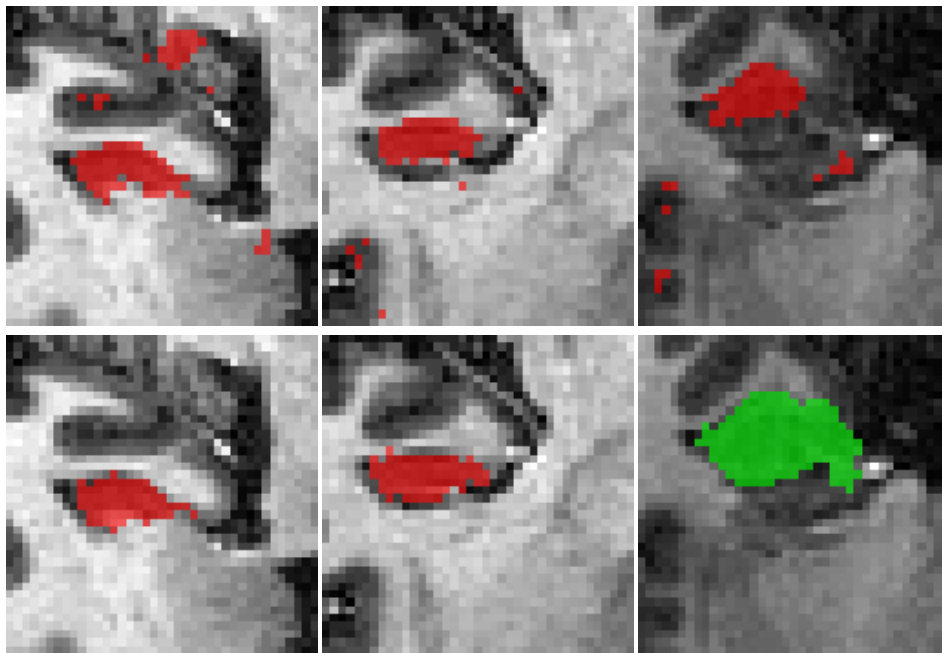


Figure 4.23: Slices of a volume from split 0 used for hippocampus segmentation. The upper row shows predictions, while the second row shows ground truth. The Dice score of this particular volume was 0.000% for anterior hippocampus class (green) and 49.690% for the posterior hippocampus class (red).

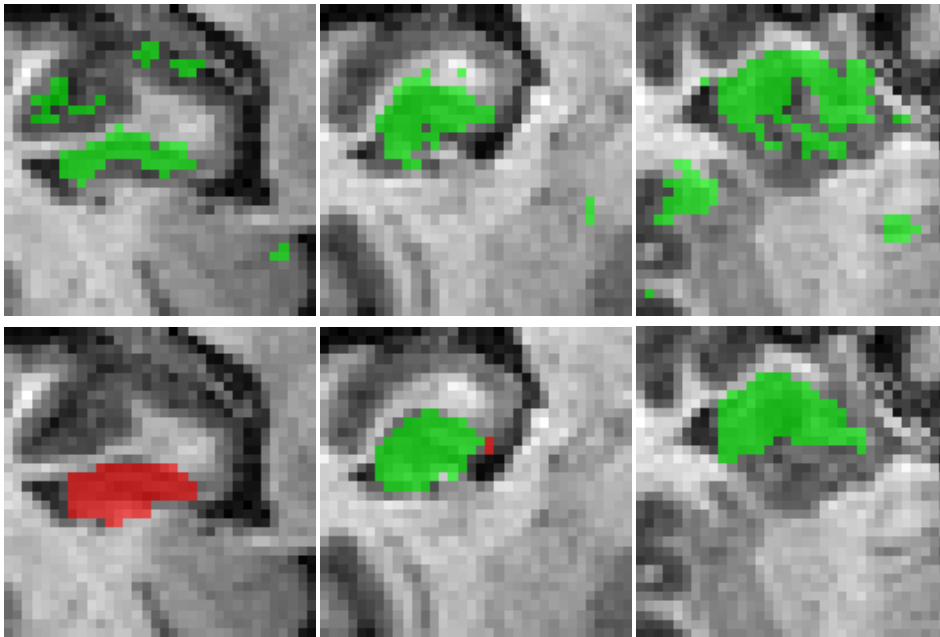


Figure 4.24: Slices of a volume from split 3 used for hippocampus segmentation. The upper row shows predictions, while the second row shows ground truth. The Dice score of this particular volume was 30.259% for anterior hippocampus class (green) and 0.000% for the posterior hippocampus class (red).

Chapter 5

Discussion

5.1 2.5D U-Net for Medical Segmentation Tasks

The 2.5D U-Net architecture was applied to several medical segmentation datasets. This addressed the first research question, **RQ1**, regarding how well the architecture would perform on medical segmentation tasks in general. Results were submitted to the Medical Segmentation Decathlon, obtaining official results comparable to state of the art.

The first experiment regarding the 2.5D U-Net used the spleen dataset for comparing the performance of using one and five consecutive input slices for predicting a single slice. The results showed slightly higher scores when using five slices as an input. Even though it could be caused by some random phenomena such as weight initialization, the performance was not worse when using five slices. Further, the use of four splits made us more confident that the improvement was not random. Intuitively, having more information along the depth axis will give better context information when predicting the segmentation. Thus, five slices were used for all the remaining experiments with 2.5D U-Net and Multi-SegCaps.

The nnU-Net has previously shown state of the art results at the Medical Segmentation Decathlon (MSD) challenge and was a well-suited benchmark to compare our results with. The test scores of nnU-Net and 2.5D U-Net were calculated by MSD using a test set without publicly available ground truth labels. Although nnU-Net achieved overall higher scores than 2.5D U-Net, we observed that some of the datasets, such as hippocampus, spleen, and liver, achieved comparable Dice scores that were within a range of only ten percentage points.

While nnU-Net used an ensemble of three models for the predictions, we used

a single model that was trained using only 75% of the training set. Ensembles have lower variance in their predictions than single model methods, and should therefore usually outperform single models [15]. Another limitation of the 2.5D U-Net framework may be that it does not allow for any dataset specific adaptations, except for the input normalization, just to keep it as general as possible. It is a fair assumption that a general algorithm not tuned for every specific dataset will produce sub-optimal segmentation results. The reason is that the method is unable to take into account any knowledge of the structure, size or location of the interesting parts of the image when preprocessing an input slice. The heart dataset does for instance only contain a small amount of its target class, roughly in the centre of the image. By training the model on an entire slice from the 3D volume, lots of more or less irrelevant background pixels in the image are considered in the loss function by the optimizer. Another consequence of this is that we get a larger class imbalance between the interesting target class and the less interesting background pixels. To compensate for the imbalance between classes, the Jaccard Dice loss is used for optimization. It inherently counteracts the effect of the imbalance, to some degree, by weighing the true positive rate of each class in relation to its precision and recall. nnU-Net used dataset specific cropping of the volumes, leaving out more of the irrelevant information and at the same time getting a lower class imbalance. This could have been an important factor for nnU-Net getting higher scores than 2.5D U-Net.

When investigating segmented volumes from 2.5D U-Net further, we observed that the model was very often capable of segmenting the rough shape of the target class, for example a heart or a spleen, but lacks pixel-perfect precision at the edges of the segmented classes. An explanation for this problem could lie in how image patches are sampled. The number of pixels belonging to a target class is calculated for every patch before using it for training, and if there are only background pixels in the slice, the slice is simply discarded. This filter ensures that there is something to segment in every slice, and is also a way of reducing the imbalance between classes. However, when there are only a few uncertain pixels belonging to the border of a target class, and tens of thousands of pixels belong to the background class, the error gradients for the few incorrectly classified pixels belonging to the target class may be lost in the vast ocean of noise coming from everything else, and thus not be able to learn to confidently classify them.

The most apparent example of the model lacking accurate segmentations at the edges is when looking at the BraTS dataset. On the surface, it appears to capture the overall shape and location of a tumorous structure but suffers confusion between the tumour classes. Looking at the precision of non-enhancing and enhancing tumour, we see scores in the low 20%, which could indicate that a lot

of the neighbouring pixels may have been incorrectly classified to those classes. While the 2.5D U-Net in this experiment used the full MRI brain images padded to a shape of 256x256, nnU-net used a heuristics for cropping the images to smaller patches of size 192x160. Because nnU-net used more preprocessing (e.g calculating number of layers according to the dataset) and postprocessing (e.g connected component analysis according to statistics about datasets) steps in its pipeline, we can not conclude that using smaller, more concentrated patches alone is what gives the model a much higher accuracy.

What is interesting about the BraTS dataset in particular, is that we have in previous research applied the exact same neural network architecture, but with a smaller patch size of 128x128 (see Appendix B). When using a smaller patch size, the performance was significantly higher. Although the scores are not directly comparable as the previous results were calculated using a BraTS specific metric for evaluation. However, the difference in scores was so high that we were not in doubt that when using the smaller patch size we achieved way better segmentations. To confirm this, segmentations were also manually inspected and compared. Thus, we believe using a smaller patch size than the full 2D slice is important for achieving good results, especially when the structure being segmented only occupies a small section of the image, while the remaining is of background pixels.

Almost all datasets segmented by 2.5D U-Net experience over-segmentation on some of the slices that are containing sole background. An example of this is the circular shape that is predicted in the top left of Figure 4.2. Upon investigation, it can be seen that the structure being incorrectly predicted in most cases have a resemblance of parts of the actual structure in question. The reason why the model is unable to distinguish some similar looking structures from the actual target classes could be that we discarded all slices containing only background pixels during training. The consequence of this design choice happens to be that the model never gets exposed to this pattern, and thus face some confusion on how to handle it when performing the actual evaluation. A more suited approach could be to rather ensure that a given percentage of all slices contain some of the target classes. This is the approach taken by nn-Unet, and may in retrospect be a better suited strategy.

Regarding the first research question, **RQ1**, we believe 2.5D U-Net was able to perform well on medical segmentation tasks in general. However, the scores were lower compared to the state of the art. We believe using a smaller patch size, along with task-specific automatic preprocessing and postprocessing would be important for improving the results of 2.5D U-Net further.

5.2 Reproduce Results from *Capsules for Object Segmentation*

The results presented in the paper by LaLonde et al. showed very good performance at the task of segmenting lungs from CT images of patients. Accompanied by the paper, the authors created a repository on GitHub containing the source code used in the implementation. The exact source code was used in this experiment in an attempt to recreate the same results. In addition, the same architecture and parameter settings were used for training two other datasets. By applying SegCaps to three datasets, **RQ2** regarding reproducibility of *Capsules for Object Segmentation* and adaptability of SegCaps to similar datasets was addressed.

The average Dice score achieved when the SegCaps model was trained using the LUNA16 dataset in this project was 98.242%, which is almost as good as the average Dice score of 98.479% that was presented in the literature. The performance gap is not likely to be attributed to some random phenomena such as weight initialization, seeing that all four training splits in the original paper achieved consistently higher Dice scores than we did in this experiment. There could be several reasons why the results differ slightly. For once, we had to make a couple of assumptions about which hyperparameters should be used.

The reconstruction loss weight was unfortunately not mentioned anywhere in the paper, which was an issue when attempting to reproduce the results of LaLonde et al. Seeing that all other hyperparameters (except learning rate) had the same default value in source code as mentioned in the paper, it was a fair assumption that the actual reconstruction weight used was the same as the source code default. When using the default SegCaps parameters, reconstruction is weighted 131.072 times more than the cross-entropy loss used for segmentation. However, when inspecting the code we found that loss weights are only passed to the Keras model compilation call when using two or more GPUs for training. If the model is compiled for training on a single GPU or on a CPU, the loss weighing is simply ignored, setting the weights of segmentation and reconstruction to be equal. We believed that this was a bug in the code, which could have caused the authors to use a different weighting of reconstruction than what they thought. The researchers behind the paper were contacted regarding this issue, but we did unfortunately not get a response. Because of the uncertainty, SegCaps was trained once using a reconstruction weight of 1 and once using a reconstruction weight of 131.072. The results showed that using equal weights outperformed using higher reconstruction slightly and that using too much reconstruction could over-regularize the model, at the cost of not learning segmentation as well. We discovered that the model

was not extremely sensitive to moderate changes in reconstruction weight, as both options produced acceptable results.

The research paper stated that an initial learning rate of 0.00001 was used for training. However, in the source code, the default learning rate was 10 times higher, i.e. 0.0001. Because the learning rate was the only parameter that differed between the source code and the paper, we thought it was worth investigating if a difference in performance between the two would be observed. After training SegCaps with both learning rates, we observed that the achieved results were about the same, and the models spent roughly the same amount of time to converge. Which one of the two initial learning rates that is used was apparently not very important for training SegCaps. This observation was further substantiated by the fact that the learning rate is reduced to 1/25 of the previous after plateauing, making the initial learning rate used by the optimizer less important.

The researchers stated that they went through the laborious task of manually examining all training images in the LUNA16 dataset, and removed all mislabelled examples. The authors did not document which images that were left out, so all images were used in our experiment. A mislabelled volume was demonstrated in Figure 4.11. The lung ground truth mask is clearly incorrect, which resulted in a very low Dice score for that particular prediction. The paper acknowledges that some of the images in the LUNA16 dataset are obviously mislabelled, but does not elaborate about why they are. LUNA16 is a dataset intended to be used for nodule detection, and not lung segmentation. The documentation for LUNA16 clearly states that the lung labels should not be used as training data for segmentation models because they are themselves automatically generated by an algorithm. SegCaps is in other words supervised by another segmentation algorithm when learning LUNA16, and thus, it is not capable of learning lung segmentation any better than the algorithm it attempts to mimic. Without studying the lung labels further and comparing them to manually created labels, we can not completely rule out the thought that the lung annotations are simplified in terms of structure and geometry compared to how manually annotated labels would look. Segmenting lungs using LUNA16 may very well be a trivial task. In fact, U-Net and Tiramisu achieved Dice scores that were only 0.03 and 0.06 percentage points lower than SegCaps (Table 2.5). This demonstrates that several methods were capable of performing an almost perfect segmentation of LUNA16, and thus the task might be a too simple benchmark for segmentation algorithms.

The possible configurations for the uncertain parameters, learning rate and reconstruction weight, were explored. These were not very likely to blame for the performance gap between reported and achieved results on LUNA16. The results

in the paper itself were reported as the median of all images for every split. The paper arguments for this with an explanation that a few of the evaluated images have much worse performance than the others. For this reason, they used median instead of average. When we did not exclude mislabelled images for evaluating the performance, we likely got even more images with exceedingly poor performance (Figure 4.12). This is expected to reduce the mean average a lot, while only slightly reducing the median average. This is consistent with the results we observed in this experiment, as we had a slightly lower median than LaLonde et al. Thus, the performance gap is likely to be caused by the fact that LaLonde et al. removed some images from the training set.

The exact same SegCaps architecture was also trained on the spleen and the heart datasets from the Medical Segmentation Decathlon too. The goal was to test if it was general enough to give competitive results on other datasets. This part of the experiment addressed the second part of **RQ2**. The spleen dataset got a Dice score of 43.310%, while the heart dataset got a score of 62.091% when averaging the median per-image Dice score of all four splits. Both results were significantly lower than the results achieved by nnU-Net and 2.5D U-Net. The result was surprising, as SegCaps had slightly outperformed a U-Net based model on the LUNA16 dataset when compared by LaLonde et al.

Applying SegCaps to the spleen and the heart datasets gave disappointing results. There could be several reasons why SegCaps achieved much better results on the LUNA16 dataset. Our initial hypothesis was that SegCaps could be extremely sensitive to hyperparameters, which possibly needed to be carefully tuned for the task at hand. However, we did experiment with some changes to learning rate, learning rate decay, and reconstruction weighting and did not observe these variations to make a significant impact on the performance. The results also showed a high variation in the score when re-training a split, which could indicate that the ability to learn was highly dependent on some random phenomena such as sampling order or weight initialization.

The spleen dataset was chosen for this experiment because it was similar to LUNA16 in several ways. The dimensions were equal and the structures to be segmented were similar, despite the spleen being of a smaller size than the lungs. This causes the spleen dataset to have a higher class imbalance, which we believed was handled by the weighted cross-entropy loss function. Table A.2 shows that lungs were only weighted about 5 times more than the background class, while spleen and heart were weighted about 70 and 145 times more, respectively. The high recall of 97.214% and 96.348% combined with a low precision of 25.494% and 43.962% for spleen and heart, respectively, could indicate that the class weights

used by the cross-entropy loss favoured the target classes too strongly. The models seemed to learn to segment most of the interesting structures, while at the same time incorrectly segmented lots of background as being part of the target class.

We were able to reproduce the results of LaLonde et al. by using the SegCaps architecture, answering the first part of **RQ2**. Despite the fact that the description of their method had some weaknesses, we believe we were able to figure out how their experiment was performed. We got a slightly lower score than LaLonde et al., which is likely due to the fact that they chose to exclude some incorrectly labelled images from the dataset. The labels of the LUNA16 dataset were themselves generated by an automatic algorithm, which had made several incorrect labellings. Due to the lack of quality control, we believe the LUNA16 segmentation dataset is insufficient for comparing algorithms. When applying the same SegCaps architecture to the spleen dataset and the heart dataset, it achieved significantly lower scores than 2.5D U-Net. To answer the second part of **RQ2**, the application of SegCaps to new datasets exposes some challenges with the architecture.

5.3 Multi-class Segmentation using SegCaps

The SegCaps implementation was in this project extended to segmentation of an arbitrary number of target classes. This was achieved by having as many capsules in the output layer as the number of classes. The assumption of extending the SegCaps architecture to a multi-class setting is that the background class itself can be represented by a single capsule in the output layer. Because the background contains a lot of variation, it was uncertain if a single capsule could be capable of learning what constitutes everything that is not of interest. Further, we were not sure if the network could scale to segmentation of multiple target classes.

By applying Multi-SegCaps to the spleen and the heart dataset it was possible to compare the architecture to the original SegCaps. For addressing whether Multi-SegCaps would be able to segment several classes, it was also applied to three datasets with multiple target classes. The goal was to answer the first part of **RQ3**, concerning the possibility of extending SegCaps and whether the Multi-SegCaps architecture would achieve similar results as the original SegCaps. The second part of **RQ3** was to compare SegCaps to a U-Net based model. While still keeping most configurations as in the experiment with the original SegCaps, some parameters were in this experiment changed to match the ones used in the 2.5D U-Net for having comparable results.

Dice loss is commonly used for training U-Net based architectures. From experience, we have seen that it ensures a stable learning process, and usually gives good results for the final segmentation. Our experience with weighted cross-entropy from the previous SegCaps experiment was that it possibly requires fine-tuned class weights and can give different results for every training despite using the same parameters and data. Testing whether a Dice loss function could be used for training a SegCaps model was therefore of interest. A Multi-SegCaps architecture was developed and trained on the spleen dataset using Dice loss. Surprisingly, it was incapable of learning how to segment anything of the target class, and all predictions were solely background; giving a Dice score of zero. The reason seems to be that the model starts to output very confident predictions for the background class and when this was combined with the non-linearity function of the output capsule, we observed that the error gradients were vanishing. This prevented the model from learning anything about the target class. As no results of interest were accomplished, weighted cross-entropy was used for the remaining Multi-SegCaps experiments.

When using Multi-SegCaps with weighted cross-entropy, the spleen and the heart datasets achieved Dice scores of 50.6% and 67.0%, respectively. The scores were a bit higher than when using regular SegCaps. We do not see any particular reason why Multi-SegCaps should achieve better results than the binary version when applied to a binary segmentation problem. However, there are some factors that could have influenced the results. When using the multi-class version of the network, the final layer consists of more capsules. The added weights in the network could possibly add some assistance in separating the positive class from the background class. We believe this is unlikely, seeing that the architectural differences are small and that segmenting background itself was not the end goal of the model, but rather a by-product of Multi-SegCaps. A more likely explanation for the increased performance is that the training routines are different. While the routine in the previous SegCaps experiments was identical to *Capsules for Object Segmentation*, the routine in this experiment was chosen to match the experiment with 2.5D U-Net, which favours shorter epochs and a smoother learning rate decay. Another factor which could have influenced the gap is the number of consecutive slices provided to the model during training. While SegCaps was only given one slice, Multi-SegCaps was given five slices. The first experiment showed that using multiple slices gave a slightly higher score, which could be a part of the explanation behind Multi-SegCaps' higher performance.

The last explanation for the gap in performance of SegCaps and Multi-SegCaps could be attributed to some random phenomena, such as the weight initialization. As for the original SegCaps, it was discovered that retraining the same splits sev-

eral times could yield completely different results. Split 2 of the spleen dataset achieved a poor test score of 21.6%, and when retraining the split it was able to gain 64.37%. Despite the unstable learning, we believe that the scores show that the extension of SegCaps works, as it achieved results for the spleen and the heart that was at least as good as the results from the original binary SegCaps.

Multi-SegCaps was further tested on the hippocampus dataset. Dice scores of about 70% were reached for both anterior and posterior hippocampus. The dataset was well suited for testing the architecture because it contains two target classes of roughly equal prevalence. In addition, the hippocampus images are relatively small, making the task fast to learn. The score was about 10% lower than for the 2.5D U-Net model, for both classes. This is a smaller gap in performance for the architectures than what was observed for the spleen and the heart datasets. The scores show that the extended SegCaps architecture was indeed able to learn two target classes. However, inspecting the hippocampus predictions revealed some confusion between the two classes, a similar issue as experienced with 2.5D U-Net. A reason could be that classes lay next to each other. Upon investigation, we were not able to see the boundary of the labels when inspecting the input images, which might mean that the boundary was arbitrarily placed by human experts and that the labels could contain some inconsistencies.

Applying the more difficult liver and BraTS datasets to Multi-SegCaps did not give any remarkable results. The model did accomplish reasonable results for the largest target class of each dataset: liver and edema. It was able to learn some of the non-enhancing tumour in the BraTS dataset, but liver tumour and enhancing tumour were close to having a Dice score of 0%. The reason for the poor performance on these datasets might be composed of several factors. Tumours are highly inconsistent in shape and appearance, making the task of segmenting them challenging. In addition to being hard to locate, the least appearing classes might not have had a high enough class weight for the weighted cross-entropy loss function. Another reason for the poor performance could be that some concepts might be difficult to represent for a capsule. If an entity, such as a liver tumour, presents itself with high variability, it could be difficult for some capsules to estimate its instantiation parameters accurately. This would cause the capsule to not learn certain concepts sufficiently. To investigate this possibility further, the dimensions of capsules could be increased. Unfortunately, it was not possible for us to test this hypothesis on the relevant datasets, due to memory limitations.

There is a possibility that the Multi-SegCaps models were not fully trained. Despite letting the models train for as many as 100 epochs after plateauing, we did experience that they could continue learning after being stuck on the same valida-

tion score for a long time. Depending on the complexity of the task, we experienced that the models would early stop after three to five days. Further research could investigate if SegCaps architectures need even more time to converge.

The memory requirements of SegCaps (and Multi-SegCaps) is an obvious limitation. Using a larger batch size would have sped up the training process and would likely have improved the results. A larger batch size would allow for more stable gradient updates, thus allowing for faster convergence. This could also have solved the previously described issue where the model gets stuck in a local minimum during training and is not able to improve before the early stop limit runs out. Multi-SegCaps was not able to learn segmentations with the same precision as 2.5D U-Net, and currently requires orders of magnitude more memory and time. Optimizing the dynamic routing algorithm to use less memory and compute time is absolutely necessary if SegCaps, and other capsule networks for that matter, is going to be used in the future.

The SegCaps architecture for multi-class segmentation obtained results as good as those for the original binary architecture. When segmenting anterior and posterior hippocampus, scores close to the ones by 2.5D U-Net were achieved. These results show that we were able to successfully extend SegCaps to multi-class segmentation, as questioned in **RQ3**. The challenges appearing when applying Multi-SegCaps to more difficult tasks is likely due to issues already existing in the SegCaps architecture. In the second part of **RQ3** we questioned whether SegCaps would be able to compete with a U-Net based model. Except for the hippocampus dataset, neither the original or Multi-SegCaps had a performance close to 2.5D U-Net in segmentation performance.

5.4 SegCaps with EM-routing

Geoffrey Hinton is a pioneer in the field of deep learning and computer vision. Lately, he has been working on capsule networks, which has resulted in two publications so far: *Dynamic Routing between Capsules* and *Matrix Capsules with EM-routing*. In the latter paper, he argues that the capsule architecture they developed is promising, seeing that it achieves very good results at certain image classification tasks while requiring relatively few trainable parameters. In this experiment, an architecture based on matrix capsules was implemented to research if matrix capsules and the EM-routing algorithm could be adapted for image segmentation, while also investigating how feasible this is. By applying the developed architecture to the hippocampus dataset, the results would be used to address the last research question, **RQ4**.

Initially, the model was developed to perform binary segmentation of the anterior and posterior hippocampus classes as a single target class. The model achieved a total Dice score of 54.5% when combining the predictions of all four training splits. Although the result itself was not very good compared to 2.5D U-Net or SegCaps, it showed that the method was indeed capable of performing segmentation, albeit on a primitive level. In Figure 4.22, we see that the model is capable of capturing the rough structure of the hippocampus, giving the model a high recall of 71.264%. Besides segmenting the hippocampus, the model shows a tendency to incorrectly segment structures of background as hippocampus as well. There might be several reasons why the model is not capable of constraining the segmentation to only the region of interest. The network has only 30,166 trainable parameters, which might be too few to learn segmentation of complex structures.

Although the low number of parameters may be partly to blame for the low performance, another reason could be that the capsule network itself is not able to accurately reason about the precise location of detected entities at different capsule layers. Due to resource demands, EM-SegCaps has relatively few layers in the network and small receptive fields of 3x3 in its convolutional capsule kernels. Hence, the model may be too simple to reason about patterns observed at one location in the image using information from other regions. Using the hippocampus dataset, it is expected that the model will learn that it is supposed to segment a single solid structure. Any other structures resembling a hippocampus should, in theory, be explained away by the presence of another more prominent hippocampus-like structure. The predictions contain multiple separate structures, as well as individual pixels, labelled as the hippocampus. The explanation could be that the network is either not able to take global feature information into account fully, or that the EM-routing algorithm fails at routing information in the way we would expect, when used for segmentation.

Finding a better suited loss function may also improve the network's ability to learn. Spread loss, which was used in the original matrix capsules paper, weighted cross-entropy and Jaccard Dice loss were tested for the architecture. None of these loss functions were able to accomplish anything besides predicting everything as background. The reason why the modified Jaccard Dice function was used in this experiment is that it was the only loss function that managed to segment anything at all. The loss function appears to be even less sensitive to class imbalance than Jaccard Dice. Thus, the choice of loss function was more or less arbitrary and is subject to improvements in the future.

For simplicity, the current EM-SegCaps implementation does not utilize the coor-

dinate addition technique, which is reported by *Matrix Capsules with EM-Routing* to slightly improve the capsules' ability to encode the spatial location of objects in an image. It is doubtful that this alone will solve the current problem of over-segmentation, as the reported improvements were only marginal. However, in the future, it would be interesting to research how significant its effect would be on segmentation tasks.

The second part of the EM-SegCaps experiment trained the architecture to segment both the anterior and posterior hippocampus classes. This was done in order to make the results comparable to the other experiments and investigate the feasibility of EM-routing for multi-class tasks. The model got a Dice score of 18.674% for anterior and 24.523% for the posterior hippocampus classes. An interesting observation is that all the correct predictions were made by either split 0 or split 3. Splits 1 and 2 predicted all pixels as background. The model did not seem to be capable of separating the two hippocampus classes from each other. While split 0 had correct predictions for only the posterior hippocampus class, split 3 predicted only the anterior class. A reason might be that there are too few capsules in the later layers, which are used for the final segmentation. This could cause an information bottleneck that prevents useful information from being used in the final evaluation.

The purpose of the experiments with EM-SegCaps was primarily to explore the possibilities of using the EM-routing technique for image segmentation, rather than creating the next state of the art framework. Matrix capsules were shown to be capable of segmenting simple structures with some results, but a lot of work still remains. When developing this architecture further in the future, a different dataset than hippocampus, consisting of simple geometry would make the results and how the model reasons more interpretable. The high memory requirements by the model currently limit how far it can scale with regards to the input image size and the number of target classes. The reason why EM-SegCaps was only tested on hippocampus data, is because slices of volumes from that dataset are only around 32x32 pixels in size. The implementation of EM-SegCaps does not use any native Tensorflow operations for the EM-routing algorithm, nor the matrix convolution operation. Because these operations are not implemented and optimized in a low-level programming language, they have to be emulated in Python code using other Tensorflow operations. This made the calculations in this experiment extremely expensive both in terms of computation and memory. According to the *Matrix Capsules with EM Routing* paper, Hinton and his associates are currently working on scaling the *Capsules architecture* to larger datasets. We believe this would require lots of optimizations to be feasible.

This experiment showed a proof of concept for an implementation of SegCaps with EM-routing. The model was able to learn binary segmentation of hippocampus, albeit on a primitive level. Our experiments showed that segmentation using SegCaps with EM-routing appears to be possible, at least on a simple dataset, answering **RQ4**. We recommend that segmentation is revisited later when the method of EM-routing has matured.

5.5 Reflections

After previously having success with 2.5D U-Net (Appendix B) on the BraTS challenge, we were excited to apply the dataset to multiple medical segmentation tasks. The scores achieved in this project were lower than the ones achieved by nnU-Net and lower than our previous results. The main difference was the patch size, as it was changed to match the one used by SegCaps. In retrospect we should probably have spent more time on optimizing automatic task-specific things as patch size, preprocessing and postprocessing. This was the focus suggested by the authors of nnU-net, who won the Medical Segmentation Decathlon with an ensemble of simple U-Net models.

During the time we worked with SegCaps some mistakes were discovered, both in the code base and in the research method. This brings out the importance of reproducing research and not trust results blindly. Although the correction did not change the performance significantly, one of the errors found in the source code did impact the result score. In addition, our experiments showed that the performance on LUNA16 was not representative of the general performance of the architecture. We emphasize the importance of using quality assured datasets when assessing the quality of a new architecture.

During this project, we could have spent time exploring other capsule network topologies, that are not based on the U-Net style encode-decoder path. Instead of spending lots of time on extending the SegCaps architecture into performing multi-class segmentation, time could have been spent on changing the architecture of SegCaps. The experiment with the original SegCaps showed unsatisfactory results and this was not the best basis for building an even more complicated architecture.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Early diagnosis and correct treatment are important for improving a patient's quality of life and increasing their chance of survival. There is a countless number of diseases and the criteria of diagnosis differ widely between them. Finding general methods for automatic analysis of diseases can reduce the time and costs required. Medical imaging is a common way of detecting diseases by examining the structures inside the body. Developing an automatic method for analyzing arbitrary medical images is therefore of great value to the medical community.

In this project, we developed a general U-Net based framework for automatic segmentation of medical images. The framework showed a promising ability to adapt when challenged with six medical segmentation tasks. U-Net is already extensively researched and finding new alternatives might be necessary for improving the state of the art. For this reason, we tested and further developed segmentation architectures based on capsule networks. Capsule networks show potential, despite requiring optimizations and continued research before being able to challenge U-Net.

6.2 Future Work

In the future, researchers should look into decreasing the memory consumption of SegCaps. We do not believe that the architecture will be able to outperform U-Net as it is, keeping in mind that the training process is a lot slower and requires way more memory. Furthermore, loss functions should be researched to speed up the learning process and increase the stability of learning. Other network topologies should be explored further, which are more carefully tuned to work with capsules.

For example, exploring architectures without a downsampling and upsampling path is interesting, as a fair amount of spatial information may be lost during these operations. The concatenation operations give some of the spatial information back, but this does not appear to be the optimal approach of routing spatial information in capsule networks, in our experience. Additionally, it could be of interest to investigate 3D capsule networks. We hypothesize that the dynamic routing algorithm will be able to efficiently make use of the added context information, by encoding 3D information into capsules. 3D medical images, in particular, could benefit from this approach, seeing that many body structures have a very different appearance across image slices.

To improve EM-routing, we suggest looking at decreasing memory usage. The resource requirements did not allow us to use larger receptive fields than 3×3 in the convolutional capsule kernels. With lower memory requirements, the complexity of the network could have been increased and the effect on segmentation accuracy could have been studied. Similar to SegCaps, the network topology might not be optimal and should be researched further. As the loss function may have a huge impact on the result, it would be interesting to develop new loss functions designed for segmentation with EM-routing. With such modifications, EM-routing could potentially be a realistic alternative to other segmentation methods in the future.

Bibliography

- [1] Abadi, M., et al., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from [tensorflow.org](https://www.tensorflow.org).
URL <https://www.tensorflow.org/>
- [2] Al-Rfou, R., et al., May 2016. Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints [abs/1605.02688](https://arxiv.org/abs/1605.02688).
URL <http://arxiv.org/abs/1605.02688>
- [3] Anderson, R. M., 1993. The gross physiology of the cardiovascular system. Racquet Press.
- [4] Armato III, S., et al., 01 2011. The lung image database consortium (lidc) and image database resource initiative (idri): A completed reference database of lung nodules on ct scans. *Medical Physics* 38, 915–931.
- [5] Badrinarayanan, V., Kendall, A., Cipolla, R., 2015. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. CoRR [abs/1511.00561](https://arxiv.org/abs/1511.00561).
URL <http://arxiv.org/abs/1511.00561>
- [6] Bottou, L., 2010. Large-scale machine learning with stochastic gradient descent. *Physica-Verlag HD, Heidelberg*, pp. 177–186.
- [7] Brenner, D. J., Hall, E. J., 2007. Computed tomography an increasing source of radiation exposure. *New England Journal of Medicine* 357 (22), 2277–2284, pMID: 18046031.
- [8] Brooks, R., October 1977. A quantitative theory of the hounsfield unit and its application to dual energy scanning. *Journal of computer assisted tomography* 1 (4), 487493.
URL <http://europepmc.org/abstract/MED/615229>
- [9] Caruana, R., Lawrence, S., Giles, C. L., 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In: *Advances in neural information processing systems*. pp. 402–408.

-
- [10] Chollet, F., et al., 2015. Keras. <https://keras.io>.
- [11] Cleeland, C. S., Bennett, G. J., Dantzer, R., Dougherty, P. M., Dunn, A. J., Meyers, C. A., Miller, A. H., Payne, R., Reuben, J. M., Wang, X. S., Lee, B.-N., 2003. Are the symptoms of cancer and cancer treatment due to a shared biologic mechanism? *Cancer* 97 (11), 2919–2925.
URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/cncr.11382>
- [12] Cole, L., Kramer, P. R., 2016. Chapter 6.4 - cardiovascular disease. In: Cole, L., Kramer, P. R. (Eds.), *Human Physiology, Biochemistry and Basic Medicine*. Academic Press, Boston, pp. 201 – 204.
URL <http://www.sciencedirect.com/science/article/pii/B9780128036990000426>
- [13] Cross, M., 1993. Elements of human cancer. geoffrey m. cooper. *The Quarterly Review of Biology* 68 (3), 472–472.
URL <https://doi.org/10.1086/418278>
- [14] Dice, L. R., 1945. Measures of the amount of ecologic association between species. *Ecology* 26 (3), 297–302.
URL <http://www.jstor.org/stable/1932409>
- [15] Dietterich, T. G., 2000. Ensemble methods in machine learning. In: *Multiple Classifier Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1–15.
- [16] Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., Zisserman, A., Jan. 2015. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision* 111 (1), 98–136.
- [17] Felsberg, M., 2017. Five years after the deep learning revolution of computer vision : State of the art methods for online image and video analysis.
- [18] Frank, S. A., Dec 2004. Inheritance of cancer. *Discov Med* 4 (24), 396–400.
- [19] Girshick, R. B., 2015. Fast R-CNN. CoRR abs/1504.08083.
URL <http://arxiv.org/abs/1504.08083>
- [20] Goodenberger, M. L., Jenkins, R. B., 2012. Genetics of adult glioma. *Cancer Genetics* 205 (12), 613 – 621.
URL <http://www.sciencedirect.com/science/article/pii/S2210776212002608>
-

-
- [21] Gordillo, N., Montseny, E., Sobrevilla, P., 2013. State of the art survey on mri brain tumor segmentation. *Magnetic Resonance Imaging* 31 (8), 1426 – 1438.
URL <http://www.sciencedirect.com/science/article/pii/S0730725X13001872>
- [22] Hayat, M. A., 2010. *Methods of Cancer Diagnosis, Therapy, and Prognosis*. Springer.
- [23] He, K., Gkioxari, G., Dollár, P., Girshick, R. B., 2017. Mask R-CNN. *CoRR* abs/1703.06870.
URL <http://arxiv.org/abs/1703.06870>
- [24] He, K., Zhang, X., Ren, S., Sun, J., 2015. Deep residual learning for image recognition. *CoRR* abs/1512.03385.
URL <http://arxiv.org/abs/1512.03385>
- [25] Hill, T., Marquez, L., O’Connor, M., Remus, W., 1994. Artificial neural network models for forecasting and decision making. *International Journal of Forecasting* 10 (1), 5 – 15.
URL <http://www.sciencedirect.com/science/article/pii/0169207094900450>
- [26] Hinton, G. E., Sabour, S., Frosst, N., 2018. Matrix capsules with em routing.
- [27] Hornak, J. P., 2017. *The basics of mri*.
- [28] Hubel, D. H., Wiesel, T. N., 1968. Receptive fields and functional architecture of monkey striate cortex. *Journal of Physiology (London)* 195, 215–243.
- [29] Hyman, B., Van Hoesen, G., Damasio, A., Barnes, C., 1984. Alzheimer’s disease: cell-specific pathology isolates the hippocampal formation. *Science* 225 (4667), 1168–1170.
URL <https://science.sciencemag.org/content/225/4667/1168>
- [30] Isensee, F., Kickingereder, P., Wick, W., Bendszus, M., Maier-Hein, K. H., 2018. Brain tumor segmentation and radiomics survival prediction: Contribution to the BRATS 2017 challenge. *CoRR* abs/1802.10508.
URL <http://arxiv.org/abs/1802.10508>
- [31] Isensee, F., Petersen, J., Klein, A., Zimmerer, D., Jaeger, P. F., Kohl, S., Wasserthal, J., Koehler, G., Norajitra, T., Wirkert, S., et al., 2018. nnu-net: Self-adapting framework for u-net-based medical image segmentation. *arXiv preprint arXiv:1809.10486*.
-

-
- [32] Jiménez-Sánchez, A., Albarqouni, S., Mateus, D., 2018. Capsule networks against medical imaging data challenges. CoRR abs/1807.07559.
URL <http://arxiv.org/abs/1807.07559>
- [33] Kauppi, T., Kalesnykiene, V., Kamarainen, J.-K., Lensu, L., Sorri, I., Ranninen, A., Voutilainen, R., Uusitalo, H., Klviinen, H., Pietil, J., 01 2007. Diaretdb1 diabetic retinopathy database and evaluation protocol. Vol. 2007.
- [34] Kingma, D. P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
- [35] Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., 2016. Jupyter notebooks – a publishing format for reproducible computational workflows. In: Loizides, F., Schmidt, B. (Eds.), Positioning and Power in Academic Publishing: Players, Agents and Agendas. IOS Press, pp. 87 – 90.
- [36] Krizhevsky, A., Sutskever, I., E. Hinton, G., 01 2012. Imagenet classification with deep convolutional neural networks. Neural Information Processing Systems 25.
- [37] Krogh, A., Hertz, J. A., 1992. A simple weight decay can improve generalization. In: Advances in neural information processing systems. pp. 950–957.
- [38] LaLonde, R., Bagci, U., 2018. Capsules for object segmentation. arXiv preprint arXiv:1804.04241.
- [39] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86 (11), 2278–2324.
- [40] LeCun, Y., Cortes, C., 2010. MNIST handwritten digit database.
URL <http://yann.lecun.com/exdb/mnist/>
- [41] LeCun, Y., Huang, F. J., Bottou, L., et al., 2004. Learning methods for generic object recognition with invariance to pose and lighting. In: CVPR (2). Citeseer, pp. 97–104.
- [42] LeCun, Y., Touresky, D., Hinton, G., Sejnowski, T., 1988. A theoretical framework for back-propagation. In: Proceedings of the 1988 connectionist models summer school. Vol. 1. CMU, Pittsburgh, Pa: Morgan Kaufmann, pp. 21–28.

-
- [43] Long, J., Shelhamer, E., Darrell, T., 2014. Fully convolutional networks for semantic segmentation. CoRR abs/1411.4038.
URL <http://arxiv.org/abs/1411.4038>
- [44] Louis, D. N., Perry, A., Reifenberger, G., Von Deimling, A., Figarella-Branger, D., Cavenee, W. K., Ohgaki, H., Wiestler, O. D., Kleihues, P., Ellison, D. W., 2016. The 2016 world health organization classification of tumors of the central nervous system: a summary. *Acta neuropathologica* 131 (6), 803–820.
- [45] Maska, M., Ulman, V., Svoboda, D., Matula, P., Matula, P., Ederra, C., Urbisola, A., Espaa, T., Venkatesan, S., Balak, D., Karas, P., Bolckov, T., tretitov, M., Carthel, C., Coraluppi, S., Harder, N., Rohr, K., Magnusson, K., Jaldn, J., Ortiz-de Solorzano, C., 02 2014. A benchmark for comparison of cell tracking algorithms (advanced access 10.1093/bioinformatics/btu080). *Bioinformatics* 30.
- [46] Mathews, J. D., Forsythe, A. V., Brady, Z., Butler, M. W., Goergen, S. K., Byrnes, G. B., Giles, G. G., Wallace, A. B., Anderson, P. R., Guiver, T. A., McGale, P., Cain, T. M., Dowty, J. G., Bickerstaffe, A. C., Darby, S. C., 2013. Cancer risk in 680 000 people exposed to computed tomography scans in childhood or adolescence: data linkage study of 11 million australians. *BMJ* 346.
URL <https://www.bmj.com/content/346/bmj.f2360>
- [47] McCulloch, W. S., Pitts, W., Dec 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* 5 (4), 115–133.
URL <https://doi.org/10.1007/BF02478259>
- [48] Menze, B. H., et al., Oct 2015. The multimodal brain tumor image segmentation benchmark (BRATS). *IEEE Transactions on Medical Imaging* 34 (10), 1993–2024.
- [49] Oliphant, T., 2006. NumPy: A guide to NumPy. USA: Trelgol Publishing.
URL <http://www.numpy.org/>
- [50] Prull, M., Gabrieli, J., Bunge, S., 01 2000. Age-related changes in memory: A cognitive neuroscience perspective. pp. 91–153.
- [51] Purves, D., 2001. Neuroscience (Book with CD-ROM). Sinauer Associates Inc.
URL <https://www.amazon.com/Neuroscience-Book-CD-ROM-Dale-Purves/>
-

dp/0878937420?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&
tag=chimbori05-20&linkCode=xm2&camp=2025&creative=
165953&creativeASIN=0878937420

- [52] Ren, S., He, K., Girshick, R. B., Sun, J., 2015. Faster R-CNN: towards real-time object detection with region proposal networks. CoRR abs/1506.01497. URL <http://arxiv.org/abs/1506.01497>
- [53] Ren, W., Shengen, Y., Yi, S., Qingqing, D., Gang, S., 2015. Deep image: Scaling up image recognition. CoRR abs/1501.02876, withdrawn. URL <http://arxiv.org/abs/1501.02876>
- [54] Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. Springer, pp. 234–241.
- [55] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., Fei-Fei, L., 2015. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) 115 (3), 211–252.
- [56] Sabour, S., Frosst, N., Hinton, G. E., 2017. Dynamic routing between capsules. In: Advances in neural information processing systems. pp. 3856–3866.
- [57] Seide, F., Agarwal, A., 08 2016. Cntk: Microsoft’s open-source deep-learning toolkit. pp. 2135–2135.
- [58] Simard, P. Y., Steinkraus, D., Platt, J. C., 2003. Best practices for convolutional neural networks applied to visual document analysis. IEEE, p. 958.
- [59] Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556. URL <http://arxiv.org/abs/1409.1556>
- [60] Simpson, A. L., et al., 2019. A large annotated medical image dataset for the development and evaluation of segmentation algorithms. CoRR abs/1902.09063. URL <http://arxiv.org/abs/1902.09063>
- [61] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15 (1), 1929–1958.

-
- [62] Suh, J. K., Lee, J., Lee, J.-H., Shin, S., jin Tchoe, H., Kwon, J.-W., Oct. 2018. Risk factors for developing liver cancer in people with and without liver disease. PLOS ONE 13 (10), e0206374.
URL <https://doi.org/10.1371/journal.pone.0206374>
- [63] Sun, T., Wang, Z., Smith, C. D., Liu, J., 2019. Trace-back along capsules and its application on semantic segmentation. CoRR abs/1901.02920.
URL <http://arxiv.org/abs/1901.02920>
- [64] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2014. Going deeper with convolutions. CoRR abs/1409.4842.
URL <http://arxiv.org/abs/1409.4842>
- [65] Veta, M., Heng, Y. J., Stathonikos, N., Bejnordi, B. E., Beca, F., Wollmann, T., Rohr, K., Shah, M. A., Wang, D., Rousson, M., Hedlund, M., Tellez, D., Ciompi, F., Zerhouni, E., Lanyi, D., Viana, M., Kovalev, V., Liauchuk, V., Phoulady, H. A., Qaiser, T., Graham, S., Rajpoot, N., Sjöblom, E., Molin, J., Paeng, K., Hwang, S., Park, S., Jia, Z., Chang, E. I.-C., Xu, Y., Beck, A. H., van Diest, P. J., Pluim, J. P., 2019. Predicting breast tumor proliferation from whole-slide images: The tupac16 challenge. Medical Image Analysis 54, 111 – 121.
URL <http://www.sciencedirect.com/science/article/pii/S1361841518305231>
- [66] Walker, B. R., Colledge, N. R., Ralston, S. H., Penman, I. D., 2014. Davidson's Principles and Practice of Medicine. Churchill Livingstone.
- [67] Xiao, H., Rasul, K., Vollgraf, R., 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. CoRR abs/1708.07747.
URL <http://arxiv.org/abs/1708.07747>
- [68] Yadav, D., B Lowenfels, A., 06 2013. The epidemiology of pancreatitis and pancreatic cancer. Gastroenterology 144, 1252–61.
- [69] Yushkevich, P. A., Piven, J., Cody Hazlett, H., Gimpel Smith, R., Ho, S., Gee, J. C., Gerig, G., 2006. User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. Neuroimage 31 (3), 1116–1128.
- [70] abi, S., Wang, Q., Morton, T., Brown, K. M., 2013. A low dose simulation tool for ct systems with energy integrating detectors. Medical Physics 40 (3), 031102.
-

URL <https://aapm.onlinelibrary.wiley.com/doi/abs/10.1118/1.4789628>

Appendix A

Additional Results

A.1 2.5D U-Net

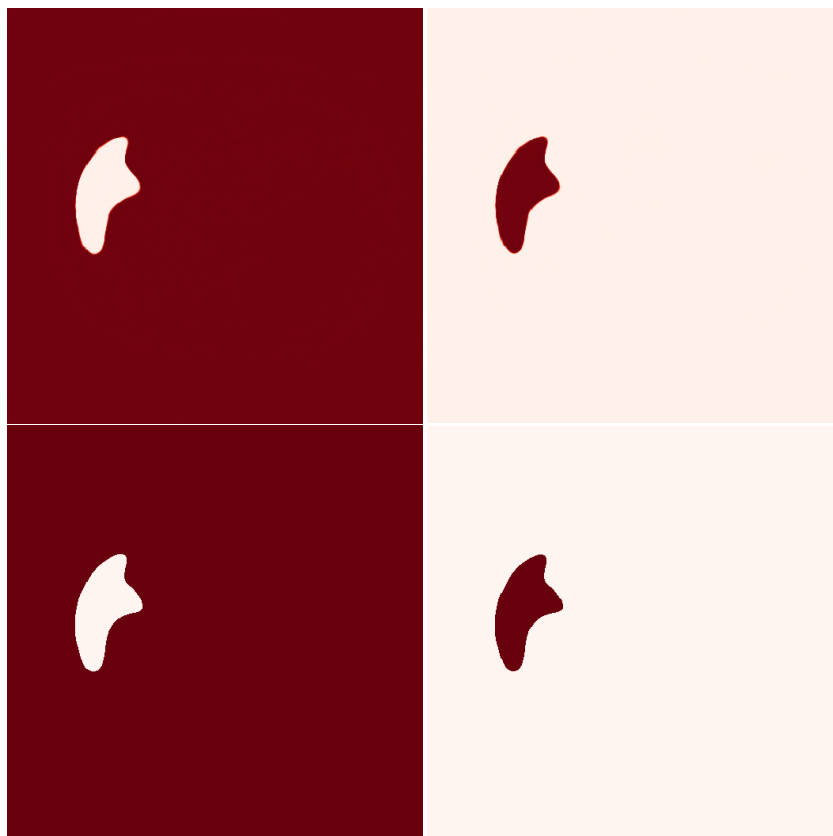


Figure A.1: Activation slice from the spleen dataset when a 2.5D U-Net model was trained using Dice loss. Top row shows network output activations, while bottom row shows expected activations. Red indicates a high output activation.

A.2 SegCaps

	Epochs (Split-0)	Epochs (Split-1)	Epochs (Split-2)	Epochs (Split-3)	Minutes per epoch
LUNA16	69	70	41	33	88
Spleen	59	50	79	56	89
Heart	82	42	29	57	39

Table A.1: Number of epochs trained on each split before early-stopping for the datasets, reported in minutes per epoch. The last column is the average training time in minutes per epoch reported using a single Nvidia P100 GPU.

Dataset	Split-0	Split-1	Split-2	Split-3
LUNA16	5.4	5.3	5.2	5.2
Spleen	69.0	69.0	69.4	70.9
Heart	145.7	150.1	139.1	146.4

Table A.2: Class weights for the different splits for SegCaps when using weighted binary cross-entropy loss

	Split-0	Split-1	Split-2	Split-3	Average
Lungs (Original results)	98.467	98.189	98.072	98.238	98.242
Lungs (Fixed labels)	98.364	98.275	98.159	98.623	98.355

Table A.3: Percentages of correct classification (Dice score) on different splits of the LUNA16 dataset before and after fixing an issue in test code. Numbers reported in median.

A.3 Multi-SegCaps

	Split-0	Split-1	Split-2	Split-3	Average
Spleen	0	0	0	0	0

Table A.4: Percentages of correct classification (Dice score) on different splits of the spleen dataset when using Dice loss. Numbers reported in mean.

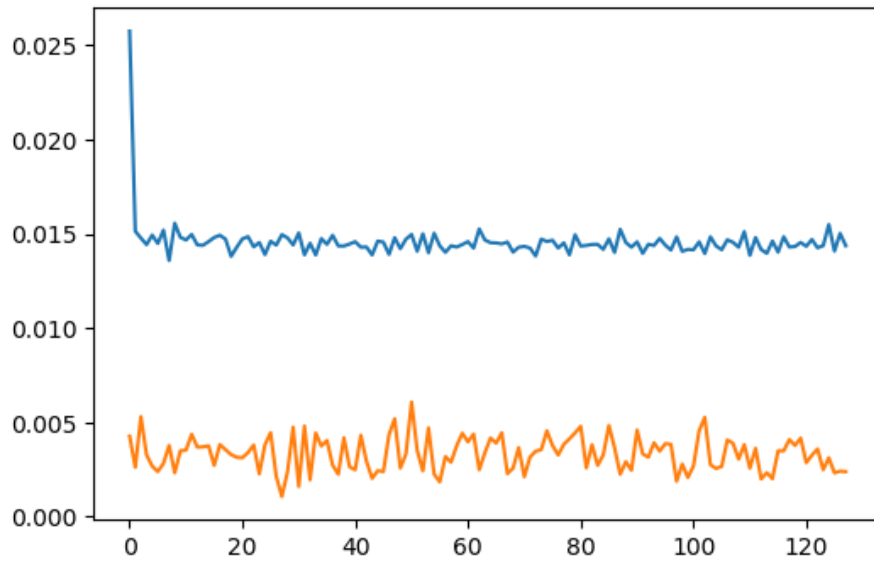


Figure A.2: Loss graph from the spleen dataset when training Multi-SegCaps using Dice loss.

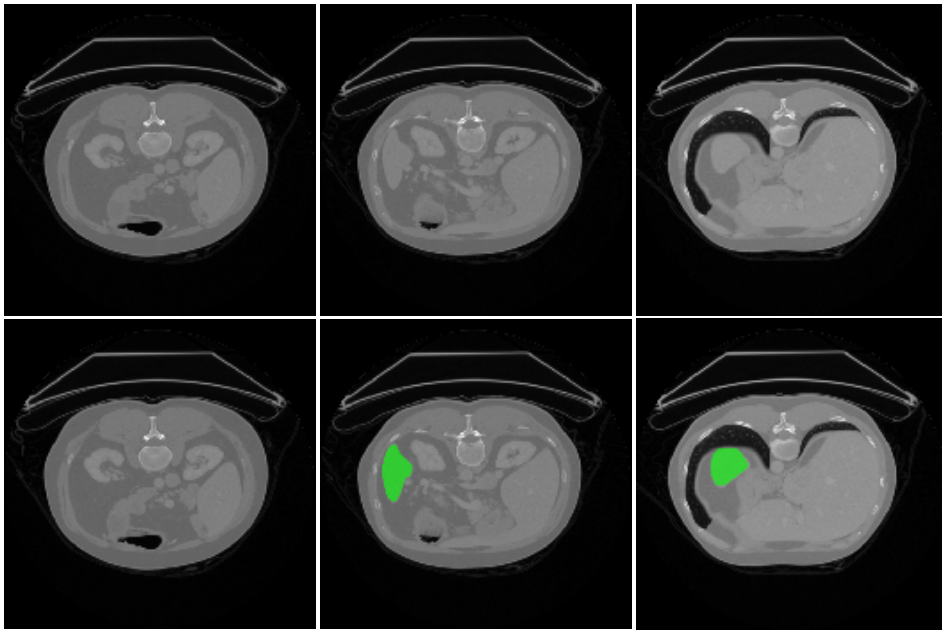


Figure A.3: Segmentation slices from spleen dataset when training Multi-SegCaps using Dice loss. Top row shows predictions, while bottom row shows labels.

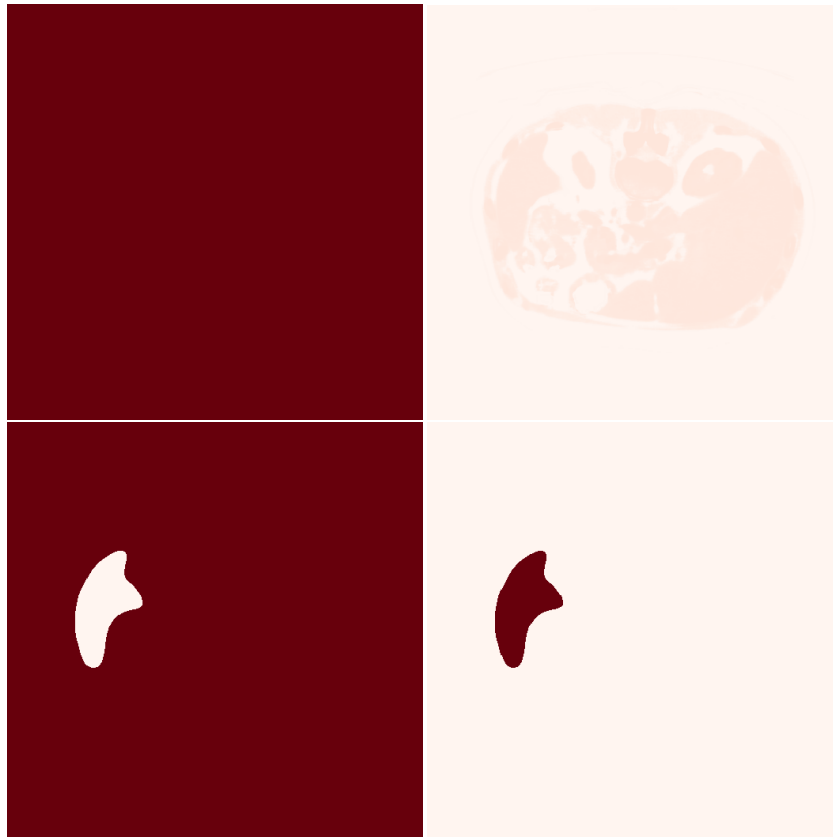


Figure A.4: Activation slice from the spleen dataset when a Multi-SegCaps model is trained using Dice loss. Top row shows network output activations, while bottom row shows expected activations. Red indicates a high output activation.

Class	Split-0	Split-1	Split-2	Split-3
Background	0.01428	0.01429	0.01421	0.01391
Spleen	0.9857	0.98571	0.98579	0.98608

Table A.5: Class weights for the different splits of the spleen dataset for Multi-SegCaps when using weighted cross-entropy.

Appendix B

NAIS Submission

A position paper was delivered for the *NAIS Symposium 2019*¹. It was accepted as a poster for the conference. The position paper was based on both previous research by the authors, as well as the work done in this thesis.

¹<https://www.aisociety.no/nais2019/>

Medical Image Segmentation using U-Net and SegCaps

Jenny Stange Johansen*, Mathias Aarseth Pedersen*, and Frank Lindseth **

Norwegian University of Science and Technology

Abstract. In recent years, automatic image segmentation using deep learning has shown great results. A CNN architecture developed by Ronneberger et al. in 2015 known as U-Net, is commonly applied to medical image segmentation[6]. Further, a new architecture called SegCaps, based on capsules networks has been introduced[3]. Even though medical image segmentation has made great strides, it is still a complex task and continuing research in this area is important.

A U-Net variant based on an article by Isensee et al.[2] was developed, as well as a SegCaps model able to segment an arbitrary number of classes. Two U-Net architectures were implemented, one for segmentation of 3D volumes, and a 2.5D architecture for segmentation of single 2D slices using four of its neighbour slices. The U-Net models were trained to perform brain tumor segmentation on the BraTS 2017 dataset. The 3D model was able to achieve very good Dice scores of 88.9%, 83.8% and 79.5% for the whole tumor, tumor core and enhancing tumor, respectively. Surprisingly, the 2.5D model had an overall better performance achieving Dice scores of 89.5%, 82.9% and 80.6%. The multi-class SegCaps model was applied to two different datasets, showing the ability to segment several classes performing reasonably well. Dice scores of 68.2% were achieved for left atrium and scores of 71.4% and 70.2% were achieved for anterior and posterior hippocampus, respectively.

Keywords: Medical Image Segmentation · U-Net · SegCaps

1 Introduction

Medical imaging is concerned with creating images of the human body. This includes both internal structures and their function. Imaging technologies are important tools for diagnosing disease and monitoring treatment response. The analysis of medical images is a time consuming task which requires expert knowledge; a type of problem that is well suited for many machine learning algorithms. Medical image segmentation attempts to extract and locate the precise location of organs, tumours and other structures of interest, with the intention of aiding health professionals in making accurate diagnoses in a shorter amount of time.

* These authors contributed equally to this work.

** <https://orcid.org/0000-0002-4979-9218>

In recent years, convolutional neural networks has been commonly used for image analysis problems. A successful application of CNN to the task of image segmentation was shown by Long et. al. in 2014, when they introduced fully convolutional networks[4]. The performance was improved by the elegant fully-convolutional U-Net architecture, published by Ronneberger et al. in the paper *U-Net: Convolutional Networks for Biomedical Image Segmentation*[6].

In 2017, Isensee et al. created a new U-Net type architecture, showing state of the art performance on the task of multimodal brain tumour segmentation using the BraTS dataset[2]. The architecture was residual and worked directly with 3D patches of medical images.

Sabour et al. presented a brand new neural network architecture in the paper *Dynamic routing with capsules* from 2017[7]. They showed that capsule networks gave state of the art performance on the task of classifying highly overlapping digits in the MNIST dataset, proving CapsNet to be an efficient architecture for learning this type of problem.

Building on the work of capsule networks, LaLonde et al. presented a method for performing binary image segmentation using capsule networks in their paper *Capsules for Object segmentation*[3]. The number of parameters needed by the architecture was reduced substantially compared to other capsule network implementations, by constraining the dynamic routing to capsules only in a defined kernel. The network showed good results when tested on lung segmentation using the LUNA16 dataset.

In 2018, Hinton et al. released the paper *Matrix capsules with EM routing*[1]. The authors claimed that the EM-routing algorithm solves most of the issues with the routing algorithm from *Dynamic routing with capsules*. It reduces the number of parameters needed, as well as using a metric of agreement that does not saturate with highly confident predictions. It gave state of the art performance at detecting objects at novel viewpoints on the smallNORB dataset.

2 Methodology, Contributions and Datasets

U-Net based models Two U-Net based models were used to study the benefits of using 3D convolutions directly on volumetric data, compared to using regular 2D-convolutions on 5 neighbouring slices of the same volume. The 3D model was based on the architecture by Isensee et al.[2], and the 2.5D model was identical, except that 3D operations were replaced with 2D operations. The replaced operations were convolutions, spatial dropout and nearest neighbour upsampling.

SegCaps based models The SegCaps architecture by LaLonde et al.[3], was extended to perform multi-class image segmentation, while also supporting multiple input slices and multiple modalities. The architecture was constructed using convolutional and deconvolutional capsule layers as building blocks in a U-Net type encoder-decoder architecture. The convolutional capsule layers attempt to

predict the outputs for the next capsule layer in the network using the locally-constrained dynamic routing algorithm.

The architecture used was identical to binary SegCaps until the last output segmentation capsule layer. In the binary segmentation architecture, the segmentation layer consists of a single 16D capsule that is shared between all pixels. The euclidean length of this capsule at locations in the image decide if a given pixel should be assigned to the background class or the positive class. In the developed multi-class SegCaps architecture, the segmentation layer has as many capsules as there are classes in the dataset, including background. The class represented by the capsule having the longest euclidean length, is simply the predicted class for a given pixel.

To extend input reconstruction to an arbitrary number of classes, only pixels labelled as not background were reconstructed. The reconstruction layers attempts to reconstruct every input modality provided. The purpose of using input reconstruction is to incentivize the network to create meaningful representations of structures in the data.

Datasets Images from the BraTS 2017 dataset[5] that was compiled for the Medical Image Decathlon competition[8] were used in the experiments. They were normalized and augmented. The dataset consists of MRI images with four modalities: Flair, t1-weighted, t1-weighted with gadolinium and t2-weighted. All image modalities were provided to the network as separate input channels.

The heart dataset was obtained from the Medical Segmentation Decathlon[8]. The goal is to segment the left atrium from one modality MRI images. The dataset was normalized and augmented.

The hippocampus dataset, was like the other datasets retrieved from the the Medical Segmentation Decathlon image collection. The hippocampus dataset has two positive classes, anterior and posterior hippocampus. The dataset was normalized and augmented.

3 Experiments and results

3.1 3D and 2.5D U-Net models

The 3D U-Net model was trained using image patches and labels of 128x128x128 voxels that were randomly sampled from BraTS training images. The 2.5D U-net model was trained using randomly sampled patches of 128x128x5 voxels, with a corresponding 128x128 pixel label belonging to the slice in the middle of the 5 provided. The other four slices were given as additional input channels to support the segmentation of the middle slice. An overview can be seen in Table 1. Both models were trained using Jaccard-Dice loss.

Table 2 gives an overview of the performance of the 3D model and the 2.5D model. The performance was measured using the Dice coefficient, recall, and precision on the different part of the segmentation. The measurements are given for the whole tumor, the tumor core and the enhancing tumor. The calculations

	Convolution type	Slice size	Epochs	Minibatch size	Image modalities
3D U-Net	3D	128x128x128	300	2	FLAIR, T1w, T1Gd, T2w
2.5D U-Net	2D	128x128x5	300	32	FLAIR, T1w, T1Gd, T2w

Table 1. An overview of U-Net models trained for brain tumour segmentation.

were made using 96 images from the BraTS 2017 dataset, which was not used during training. An example of segmentation from the 2.5D U-Net can be viewed in Figure 1.

Model	Dice			Precision			Recall		
	Whole	Core	Enh.	Whole	Core	Enh.	Whole	Core	Enh.
3D U-Net	88.9	83.8	79.5	94.3	88.8	78.8	84.0	79.3	80.1
2.5D U-Net	89.5	82.9	80.6	90.3	87.5	80.5	88.7	78.7	80.6

Table 2. Results on validation data for the U-Net models showing percent Dice scores.

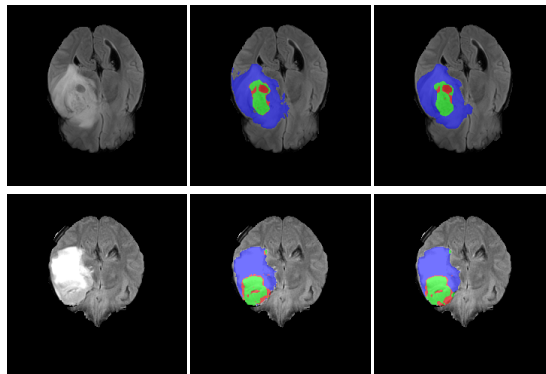


Fig. 1. Segmentation of two images using 2.5D U-Net. From left to right: FLAIR image, ground truth and prediction. Edema, enhancing tumor and non-enhancing tumor, marked in blue, green and red, respectively.

3.2 Multi-class SegCaps

The multi-class SegCaps model was trained on the heart dataset and the hippocampus dataset. Both experiments used four splits for training. The loss

functions used were weighted binary cross-entropy for segmentation and mean squared error for reconstruction. The training was performed using a batch size of 1, due to memory limitations.

Heart The performance of multi-class SegCaps trained on the heart dataset can be seen in Table 3. The experiment shows fairly good results, but the performance has a high variance. An example segmentation can be seen in Figure 2.

	Split-0	Split-1	Split-2	Split-3	Average
Left Atrium	68.0	60.9	77.2	66.8	68.2

Table 3. Percentages of correct classification (percent of dice score) on the mean average for different splits of the heart dataset.

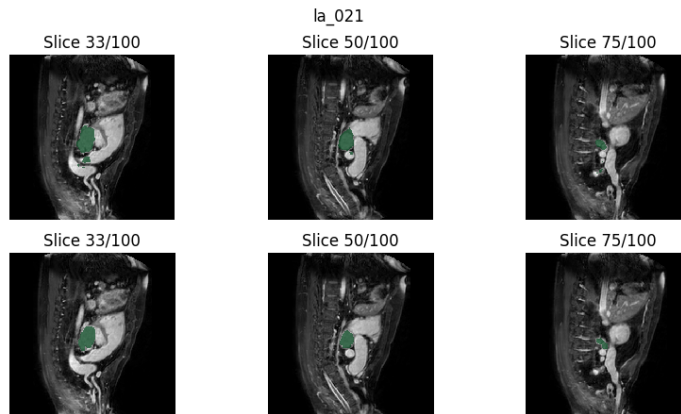


Fig. 2. Slices from an example image from heart segmentation. The upper row shows predictions, while the second row shows ground truth. The score of this particular volume was 76.8%.

Hippocampus Table 4 shows the result of training multi-class SegCaps on the hippocampus dataset. The Dice scores show that the architecture is able to segment two classes with high performance. An example of segmentation can be seen in Figure 3.

	Split-0	Split-1	Split-2	Split-3	Average
Anterior	69.8	70.4	73.6	72.0	71.4
Posterior	69.7	67.8	68.1	75.2	70.2

Table 4. Percentages of correct classification (percent of dice score) on the mean average for different splits of the hippocampus dataset.

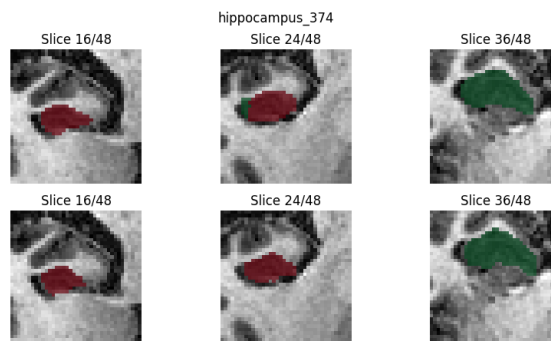


Fig. 3. The upper row shows predictions, while the second row shows ground truth. The dice score of this particular image was 76.5% for anterior hippocampus class (green) and 79.1% for the posterior hippocampus class (red).

4 Conclusion and future work

For the 3D and the 2.5D U-Net architectures, the differences to Isensee et al. were minor, yet we achieved slightly better Dice scores than they did. There might be several reasons for this, including the use of a larger subset of the BraTS dataset for training, a larger amount of elastic deformations and using a different loss function. Surprisingly, the model we developed for 2.5D segmentation achieved a higher Dice score than the 3D model on both whole tumor and enhancing tumor. This could mean that global context information is less important for tumor segmentation than we initially thought.

A CapsNet architecture was developed showing its ability to segment several output classes. The performance on the datasets was lower than state of the art results for U-Net models[8]. However, we believe that the results were promising. In the future, modifications allowing of a lower memory consumption could be interesting to look at. This would allow for a larger batch size and more efficient training. We continue our research by applying the architecture to the BraTS dataset, which might give more insights to the future of multi-class segmentation by using SegCaps. Further we are also looking at EM-routing for SegCaps, which could allow for segmentation using drastically less parameters.

References

1. Hinton, G.E., Sabour, S., Frosst, N.: Matrix capsules with em routing (2018)
2. Isensee, F., Kickingereder, P., Wick, W., Bendszus, M., Maier-Hein, K.H.: Brain tumor segmentation and radiomics survival prediction: Contribution to the BRATS 2017 challenge. CoRR **abs/1802.10508** (2018), <http://arxiv.org/abs/1802.10508>
3. LaLonde, R., Bagci, U.: Capsules for object segmentation. arXiv preprint arXiv:1804.04241 (2018)
4. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. CoRR **abs/1411.4038** (2014), <http://arxiv.org/abs/1411.4038>
5. Menze, B.H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., Burren, Y., Porz, N., Slotboom, J., Wiest, R., Lanczi, L., Gerstner, E., Weber, M., Arbel, T., Avants, B.B., Ayache, N., Buendia, P., Collins, D.L., Cordier, N., Corso, J.J., Criminisi, A., Das, T., Delingette, H., Demiralp, ., Durst, C.R., Dojat, M., Doyle, S., Festa, J., Forbes, F., Geremia, E., Glocker, B., Golland, P., Guo, X., Hamamci, A., Iftekharuddin, K.M., Jena, R., John, N.M., Konukoglu, E., Lashkari, D., Mariz, J.A., Meier, R., Pereira, S., Precup, D., Price, S.J., Raviv, T.R., Reza, S.M.S., Ryan, M., Sarikaya, D., Schwartz, L., Shin, H., Shotton, J., Silva, C.A., Sousa, N., Subbanna, N.K., Szekely, G., Taylor, T.J., Thomas, O.M., Tustison, N.J., Unal, G., Vasseur, F., Wintermark, M., Ye, D.H., Zhao, L., Zhao, B., Zikic, D., Prastawa, M., Reyes, M., Leemput, K.V.: The multimodal brain tumor image segmentation benchmark (BRATS). *IEEE Transactions on Medical Imaging* **34**(10), 1993–2024 (Oct 2015). <https://doi.org/10.1109/TMI.2014.2377694>
6. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: *International Conference on Medical image computing and computer-assisted intervention*. pp. 234–241. Springer (2015)
7. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. CoRR **abs/1710.09829** (2017), <http://arxiv.org/abs/1710.09829>
8. Simpson, A.L., Antonelli, M., Bakas, S., Bilello, M., Farahani, K., van Ginneken, B., Kopp-Schneider, A., Landman, B.A., Litjens, G.J.S., Menze, B.H., Ronneberger, O., Summers, R.M., Bilic, P., Christ, P.F., Do, R.K.G., Gollub, M., Golia-Pernicka, J., Heckers, S., Jarnagin, W.R., McHugo, M., Napel, S., Vorontsov, E., Maier-Hein, L., Cardoso, M.J.: A large annotated medical image dataset for the development and evaluation of segmentation algorithms. CoRR **abs/1902.09063** (2019), <http://arxiv.org/abs/1902.09063>