Semi-supervised Lesion Detection via Image Restoration

Master's Thesis

Jonatan Kronander

Supervisors:
Xiaoran Chen
Prof. Dr. Ender Konukoglu
Prof. Dr. Kalle Åström

August 31, 2020

Abstract

Lesion detection is a critical task for medical image understanding. While the problem has been widely addressed in a supervised semantic segmentation manner, the problem clinically appears more similar to novelty detection with few or no annotations for the lesion. The reason is two-fold: 1) it is intuitively easier to collect large dataset from healthy individuals than that from a specific type of lesion individuals, 2) clinicians are generally interested in any abnormalities regardless of its type. This makes unsupervised methods more attractive solutions. Works such as AnoGAN and VAE with image restoration offer practical ways to localise lesions by training only on healthy data. However, for the same type of lesion, an obvious performance gap exists between unsupervised and supervised methods. In this work, we intend to provide supervision with a small number of lesion data to the unsupervised method with the aim to narrow the gap. The method is an extension of the unsupervised method of VAE with MAP-based image restoration. In more details, we train an U-Net on the few examples to predict the likelihood term and impose the supervision with annotated lesions such that the restoration only occurs for the lesion pixels. We train the unsupervised method on T2-weighted images of healthy individuals of Cam-CAN dataset and provide a small annotated dataset consisting of a few subjects from BraTS dataset, and test on an unseen subset of BraTS. With the addition of the few examples, the method shows an improvement over the unsupervised method while the gap with the supervised is narrowed but still exists.

Acknowledgements

I want to thank Xiaoran Chen and Prof. Dr. Ender Konukoglu for their continuous support, from beginning to end. Thank you for making this thesis possible, answering all my questions and guiding me in the right direction. I have learned lots of both domain knowledge and how research is conducted and I am forever grateful for that. I am also thankful to Prof. Dr. Kalle Åström who made this thesis possible from Lund University. This project would not have been possible without the support of Lund University and ETH Zurich, who provide a great program for exchange students.

Acronyms

AUC	area under the ROC curve
Cam-CAN	Cambridge Centre for Ageing and Neuroscience
CNN	convolutional neural network
ELBO	evidence lower bound
FLAIR	fluid attenuated inversion recovery
GMVAE	Gaussian mixture variational autoencoders
GAN	generative adversarial network
KL	Kullback-Leibler
MRI	magnetic resonance imaging
BraTS	Multimodal Brain Tumor Segmentation Challenge
MAP	maximum a posterior probability
TV norm	total variation norm
VAE	variational autoencoder

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Lesion detection can be seen as outlier detection, where outlier observations, expensive to accommodate in a prior model, are detected. One common approach for lesion detection is to learn prior distributions from unlabelled data in an unsupervised framework. However, in most practical scenarios there are often additionally small amounts of labelled lesion data available. Such a scenario often occurs in medical imaging, where clinicians often have large amounts of unlabelled healthy subjects available, but due to the tedious task of manual annotation only performed by trained domain experts, only a few labelled abnormal subjects available. The labelled abnormal/lesion data, combined with unlabelled healthy subjects, could potentially be used to increase performance of the unsupervised models by extending these into semi-supervised models. The problem of automatic brain tumour segmentation is a good example where usually there are many unlabelled healthy images available, but only a limited amount of labelled images of tumours.

Detecting a lesion, like a brain tumour, is an important and daily task for radiologists. Today, there are already many satisfying automatic segmentation models being proposed. However, most of these models are learned in a supervised framework and require lots of labelled data. Additionally, these models are often restricted by how well the training data generalises the problem and these models typically fail to detect a tumour that is very different from the training data. For example, using a supervised model for tumour detection, trained on data from other institutions and machines, typically leads to bad performance. Instead, a semi-supervised lesion detection method, as proposed in this work, constrained by using unlabelled healthy data and a limited amount of labelled data, potentially could work well for segmenting brain tumours in clinical practice.

1.1 Clinical Background

A patient diagnosed with gliomas, a type that accounts for 70% of all brain tumours, has a life expectancy of only a couple of years [44]. Approximately 2 % of all cancer diagnoses are primary malignant brain tumors and the lifetime risk of being diagnosed is 0.58 percent according to a study from Michigan State University in the United states [13]. Due of the high mortality rate, despite low prevalence of less than 0.1% [19], a brain tumour diagnosis is feared and considered very serious. There are several different types of brain tumours such as already mentioned glioma and also meningioma and pituitary tumours. They all are the result of abnormal growth of cells in the brain, called neoplasms. Each of the different types have different prognosis and treatment. The WHO (World Health Organization) is grading brain tumours from grade 1 to 5 depending on malignancy. Grade 1 gliomas are considered to be curable and are less malignant. Grade 5 gliomas are very aggressive, often removed with surgery and limit the patients life expectancy to

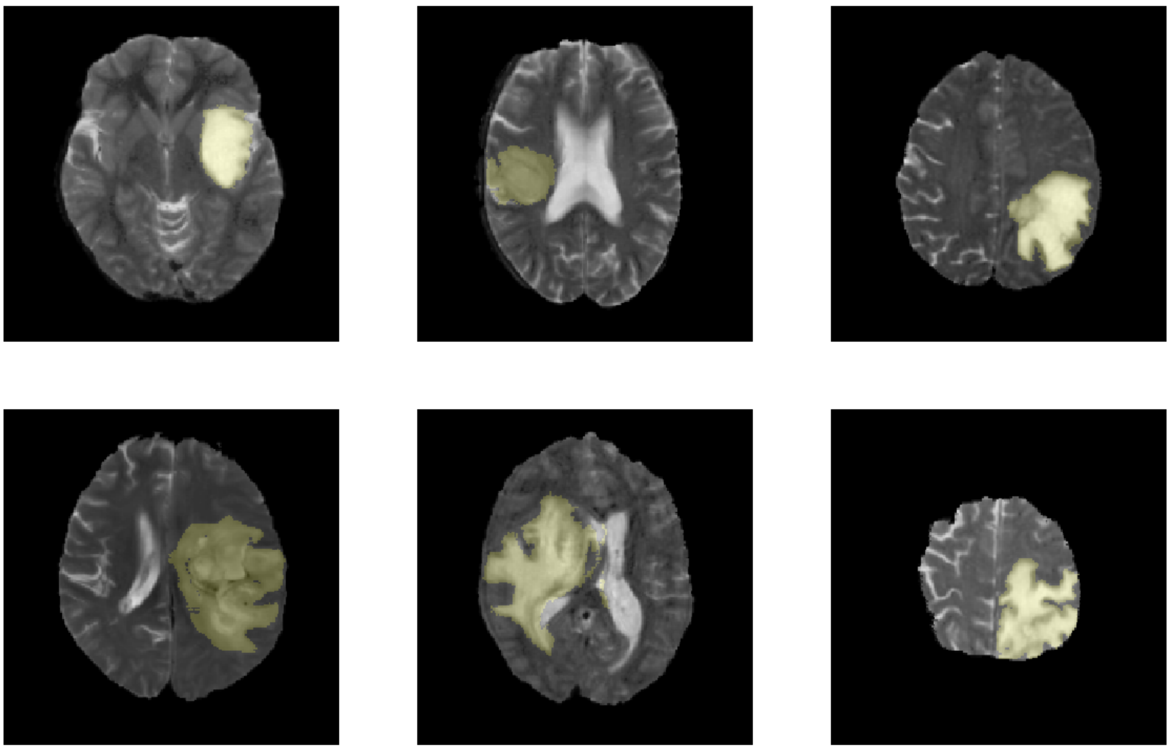


Figure 1.1: Examples of different brain scans in MRI (T2 weighted in axial plane) with tumour highlighted.

approximately 2 years after diagnosis [58].

The high mortality rate and aggressiveness of certain brain tumours makes early, fast and correct diagnosis important. Early and fast diagnosis is crucial for early treatment, which could reduce the risk of tumour growing and hence prolonging life expectancy for patients. Brain tumours are usually diagnosed using imaging, preferable magnetic resonance imaging (MRI), followed by histopathology to confirm the diagnosis [3]. Image diagnostics of brain tumours are also important for surgery and radiotherapy planning, tumour analysis, survival prediction and follow-up care.

The varying biological tissues of brain tumours consist of different histologic sub-regions such as necrotic core, peritumoral edematous, invaded tissue and gross abnormalities. In MRI, these different tissues are causing varying intensity, contrast and shape profiles. As a result of the varying profiles, every brain tumour has a unique appearance in imaging, causing diagnosis of gliomas with imaging to be considered a challenging task [7]. As seen in Figure 1.1, brain tumours appear in many forms, shapes and intensities in MRI. Pixel intensities of brain tumours, in MRI, are not easily distinguished from intensities of healthy tissue. This can be seen in Figure 1.2, that shows a histogram of healthy and tumour tissue. Additionally, brain tumours have varying spatial location and can be located in all parts of the brain, as seen in the heatmap of brain tumors in Figure 1.3, .

MRI can produce several different sequences of the same subject, by varying excitation and repetition times. Each sequence highlights different tissue types, which results in different intensities of the same tissue from different sequences. To better identify the different sub-regions in brain scans, different MRI sequences

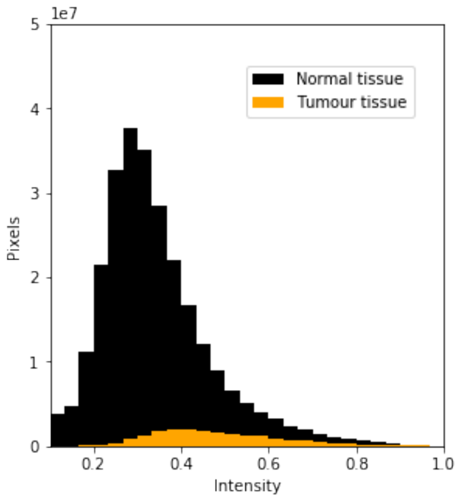


Figure 1.2: Histogram with normal brain tissue and tumour tissue pixel intensity (T2 weighted, axial slice).

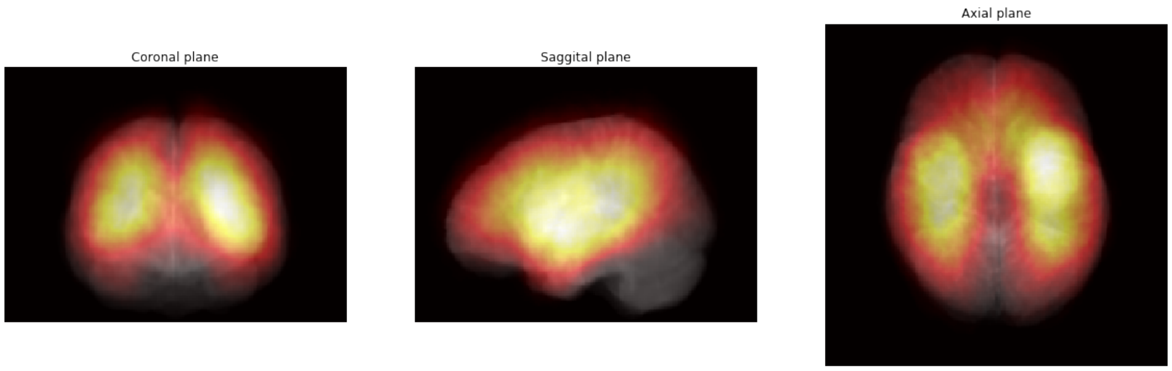
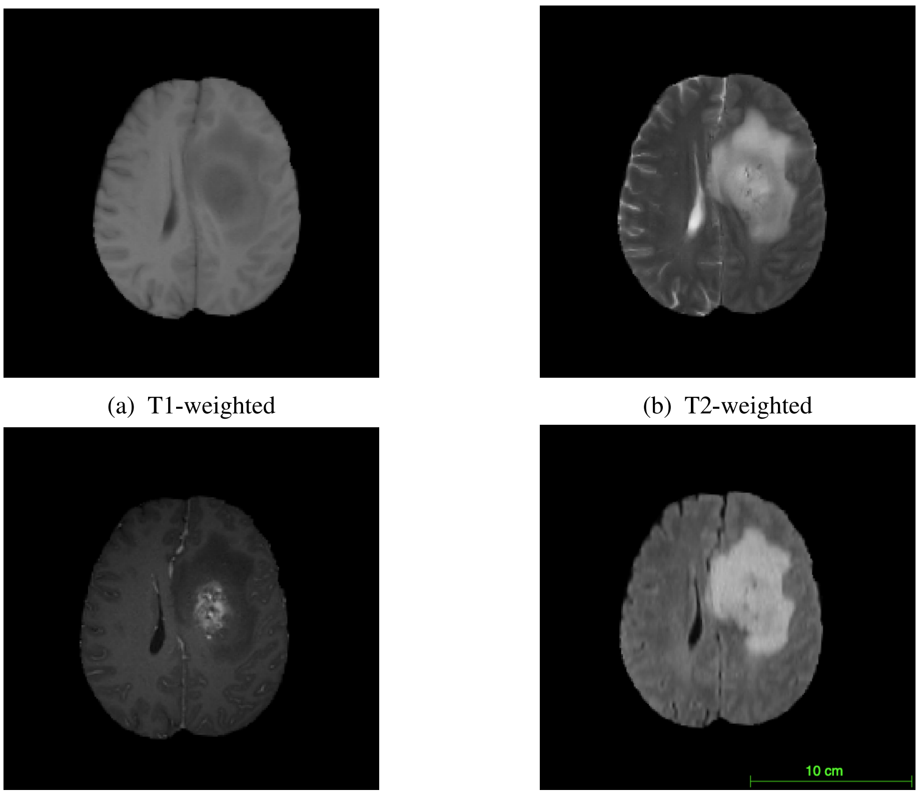


Figure 1.3: Spatial heatmap of brain tumours in BraTS dataset [40].

are often used together. Normally, physicians use the sequences T1-weighted, T1-weighted with contrast agent, T2-weighted and fluid attenuated inversion recovery (FLAIR) in MRI diagnostics [2]. Typically, the T2 weighted sequence best highlights the tumour core, whilst FLAIR highlights the whole tumour best. This is shown in Figure 2.2, that visualises one axial slice of the same brain tumour in different MRI sequences.



(a) T1-weighted (b) T2-weighted

(c) T1-weighted with contrast agent (d) Fluid Attenuated Inversion Recovery (FLAIR)

Figure 1.4: Axial slices of the same brain tumour in different MRI sequences.

1.2 Previous Work on Brain Tumour Segmentation

To help physicians diagnose and analyse brain tumours, many automated image segmentation models have been proposed by the research community. An automated image segmentation model aims to automatically find a pixel classification such as $f : X \rightarrow S$, where $X \in \mathrm{IR}^{\mathrm{N}}$ is an image and $S \in \{0, ..., C\}^N$ is the corresponding pixel wise label map. Here, N denotes the number of pixels and C denotes the number of label classes. For the semantic binary segmentation problem of whole brain tumour segmentation, S only consist of two labels $S = 1$ or $S = 0$. Either a pixel is considered as a tumour or not. If a model aims to find different sub-regions of the tumour, then each sub-region corresponds to its specific class $\in 0, ..., C^N$.

A segmented brain tumour could help improve diagnostics, growth analysis, surgical or radiotherapy planning and follow-up care. Even though there have been huge leaps in medical image analysis lately, following the achievements within deep learning, brain tumour segmentation is often still done manually in clinical practice. One reason for this is due to many methods inability to generalise outside of the training data. However, with a good automated segmentation model, physicians would not only save valuable time, but also increase diagnostic accuracy and create more reproducible measurements of tumours.

1.2.1 Tumour Segmentation in a Pre Deep Learning Era

Early work for automatic brain tumour segmentation dates back three decades, a pre deep learning era. Shad et al. [53] propose, as early as 1993, a statistical approach using clustering with texture patterns for brain tumour segmentation. Following in 1995, Vaidyanathan [61] also used a statistical approach by applying a k-nearest neighbor and spectral fuzzy C-means method. There are also more recent statistical approaches using classification or clustering, without the use of deep learning, such as Zacharaki and Bezerianos (2012) [31] and Erus et al. (2014) [24]. They used image patches and principal component analysis (PCA) to detect tumours as anomalies from healthy tissue. Gibbs et al. (1996) [25] propose, in contrast to the named statistical approaches, a deterministic approach using a growing edge detection method. Additionally, there are numerous more methods proposed to solve brain tumour segmentation, not using deep learning, that are not named here. Some of them are reviewed by Angelini et al. (2007) [5] and Bauer et al. (2013) [8], who both wrote good survey papers before the deep learning era. We refer to these papers for further reading about pre deep learning methods for brain tumour segmentation.

1.2.2 History of Deep Learning

Deep learning has over the past decade revolutionised many fields of research and industries, including medical imaging. During the past decade, research around deep learning has exploded, resulting in many achievements and new methods being proposed. Already back in 1971, Ivakhnenko [30] wrote a paper that pioneer what would later be called deep learning. LeCun et al. (1989) [36] pioneered deep learning within Computer vision by proposing to use backpropagation to train a convolutional neural network (CNN) to recognise handwritten digits. However, it would take another two decades for the disruptive technique to take off.

Even though the reader is assumed to be familiar with deep neural networks and CNNs, a short introduction is given before continuing. Deep neural networks consist of sequentially sets of non linear and linear functions with trainable parameters Θ as $\mathrm{NN}(X;\Theta) = f_0 \circ f_1 \circ f_2 \circ ... \circ f_n$ [26]. Normally deep neural networks are referred to as having an input layer f_0, followed by a number of hidden layers $f_1, ..., f_{n-1}$ and an output layer f_n. Today, there are numerous different linear and nonlinear functions and architectures proposed for different deep neural networks. One being CNNs, a specialised kind of neural networks, that uses convolutional operators as function f. CNNs has been proven to be very successful in fields like computer vision, medical imaging and time series analysis. The convolutional operator is defined as in Equation 1.1 where x is referred to as the input, w is referred to as the kernel (with learnable shared weights) and the result $s(t)$ are referred to as the feature map. The advantages of using a CNN is that each layer in the deep neural network have shared weights, additionally the convolution is shift invariant, resulting in the output not dependent on spatial shifts in the input. It has been shown that CNNs are capable of learning complex features that help the predictions of deep neural networks [65]. CNNs are often implemented in code by using fast Fourier transform (FFT) to be able to use matrix multiplication instead of a sliding window approach, which allows backpropagation to be used for training.

$$s(t) = (x * w)(t) = \sum_{x=-\infty}^{\infty} x(a)w(t-a) \tag{1.1}$$

Deep neural networks are trained and optimised to find appropriate weights for minimising a given loss function. The optimisation problem is often solved with various gradient descent methods such as stochastic gradient descent [50], RMSprop [52] and Adam [32]. Gradient descent methods require gradients with respect to parameters of the network which is normally computed by backpropagation. Hence, every function of a deep neural network optimised in this fashion needs to be differentiable. Appropriately, deep neural networks should also have good gradient flow to find the optimum parameters for a given training set.

Back to previous work within deep learning, after LeCuns work in 1989, it would take almost another two decades until deep neural networks would be widely used and achieve superhuman accuracy in computer vision problems. In 2012, Ciregan et al. [18] proposed an image classification model outperforming humans for classifying traffic signs and handwritten digits. Today, deep learning is an important tool for many research fields such as computer vision, natural language theory, machine translation, bioinformatics and medical image analysis.

1.2.3 Tumour Segmentation using Supervised Deep Learning

In a supervised learning framework, a set of training examples are available as $\{(X_1, Y_1), ..., (X_n, Y_n)\}$. In the semantic binary segmentation problem, $X_i \in \mathbb{R}^{\mathrm{N}}$ corresponds to the input image with N pixels and $Y_i \in \{0,1\}^N$ is the ground truth binary pixel wise segmentation. Supervised deep neural networks aim to find a good approximation for $f : X \to Y$, such that f also generalise well outside of the training set.

Last decade, new achievements within deep learning and deep neural networks have paved the road for many novel methods for brain tumour segmentation. Some early proposed methods for automatic brain tumour segmentation, in a supervised framework, used deep neural networks for feature extraction and classification. Two examples of this approach are, Singh et al. (2012) [55] and Amin et al. (2012) [4], who both propose methods using principal component analysis (PCA) for feature extraction but have different approaches for classification task, where support vector machines (SVM) contrary to multi-layer perceptions are used.

Training deep neural networks typically requires lots of training data to achieve good results. This is due to the millions of parameters being optimised, in addition to make the deep neural network generalise well outside the training set. In medical imaging, labelling training data for segmentation is a time intensive and a tedious task that can only be performed by trained domain experts. In addition, to create a good ground truth training set often several manual segmentations must be performed by different experts, due of the high interobserver variability between experts [7]. This makes data in medical imaging expensive and usually medical image datasets are relatively small.

Early proposed methods that use deep learning, often have small private datasets and use a variety of different metrics for measuring performance. This makes comparing these different methods for the same problem difficult. In order to solve the issues of small private datasets and incomparable metrics and push state of the art, Medical Image Computing and Computer Assisted Interventions (MICCAI) organised the public research challenge Multimodal Brain Tumor Segmentation Challenge (BraTS), in 2012. The contestants were provided a relatively big labelled dataset and the performance of the submitted methods were measured with the same metric. BraTS challange 2013 and 2015, S. Pereira et al. (2016) [47] achieved first and second place, by using a 2d patch wise CNN. Dong et al. (2017) [22] applied an U-Net architecture, proposed by Ronneberger et al. (2015) [51] (further reading in Section 2.3), on brain tumour segmentation problem with good results, scoring high in BraTS 2015 challenge. Today, while state of the art methods like from Myronenko (2018) [42], winner of BraTS 2018, gets more and more complicated, Isensee et al. (2018) [29] shows that a well trained U-Net still achieves competitive results. Myronenko and Isensee et al. both achieve over 0.9 mean dice for segmenting the whole tumour on BraTS test dataset. BraTS challenge 2017 training dataset consists of 285 3d brain scans [7], all with T1-weighted, T1-weighted with contrast agent, T2-weighted and FLAIR sequences.

1.2.4 Tumour segmentation using Unsupervised Deep Learning

Unsupervised learning provides a solution to the large amount of labelled training data required in a supervised learning framework by approximating distributions of unlabelled data. There are many techniques for unsupervised learning such as clustering, with K-means and mixture models, autoencoders, adversarial networks, sparse dictionary learning and generative models.

For example in unsupervised learning for brain tumour segmentation, a probability distribution of healthy brain images, or features of such, can be learned and tumours can be detected as outliers from these healthy distributions. This assumes that features of healthy tissue can be learned and features of lesions can be distinguished from non lesion features. In addition to not requiring annotated datasets, unsupervised methods like these also allow for detection of any brain lesion regardless if it is represented in the training dataset or not. However, to learn good distributions, in latent space, of data with high dimensionality, e.g. images, has proven to be quite challenging. So far, supervised learning methods have been outperforming unsupervised learning methods by good margins, especially for the problem of semantic segmentation. However, recently there have been many interesting methods proposed that bridge this gap in performance.

Recent success within generative models, like variational autoencoder (VAE) [33] (further reading in Section 2.1) and generative adversarial network (GAN) [27], allow for learning more complex data distributions for unlabelled data, paving the way for new unsupervised segmentation methods. Recently proposed methods for brain tumour segmentation, such as Chen and Konukoglu (2018) [15], Baur et al. (2019) [10] and You et al. (2019) [64], have been using these generative models to outperform earlier unsupervised methods.

Both Chen and Konukoglu and Baur et al. propose to first learn distributions from healthy brain images and secondly detect brain lesions as outliers from the learned distributions. The first step, to learn prior distributions, is done by either different kinds of autoencoders or GANs. The detection is then performed by reconstructing images containing brain lesions and assuming the lesion segmentation maps as the reconstruction error, i.e. the difference between the original and reconstructed image. The idea is that the learned prior model can not faithfully reconstruct regions with lesions and therefore will these regions have a larger reconstruction error. Even though these methods achieve promising results, often a large number of false positives is predicted. This might come from the generative models failing to learn rich priors for detailed features, causing high reconstruction error also for healthy tissue.

You et al. approach the problem of brain lesion segmentation by casting the segmentation problem into an image restoration problem. The restoration aims to restore lesion regions to corresponding pseudo "healthy" regions. The absolute difference between the restored and the original image is then assumed to be the lesion. You et al. solves the restoration problem with maximum a posterior probability (MAP) estimation and Bayes theorem. Before the restoration, a normative prior is learned, through a set of healthy images, with VAE [33] or the extended version, Gaussian mixture variational autoencoders (GMVAE) [21]. In the restoration, the likelihood term is approximated with total variation norm (TV norm), assuming brain tumours to have sparse gradients. An in depth description of this method can be found in Section 2.2.

Chen and Konukoglu points out that unsupervised methods, similar to mentioned above, can be interesting for a couple of reasons. Firstly, the way these models mimic human learning is itself an interesting research topic. Secondly, even previously unseen lesions are easily detected, without any need of an extensive labelled training dataset of specific lesions. Lastly, these models are an essential step to develop further semi-supervised methods that could produce considerable improvements by leveraging both labelled and unlabelled data.

1.2.5 Tumour Segmentation using Semi-supervised Deep Learning

In a semi-supervised framework there are both labelled and unlabelled data available during training. In practice, there are often large amounts of unlabelled data and a smaller amount of labelled data. Two examples of early non deep learning semi-supervised methods for brain lesion detection is Meier et al. (2014) [39] who adopted a semi-supervised decision forest and Wang et al. (2014) [62] who propose an interactive segmentation called 4D active cut.

Semi-supervised learning using deep neural networks has been proven to be successful for various tasks [45]. However, proposed methods for semi-supervised brain lesion segmentation are hard to find. Other methods applied on other segmentation problems, especially within medical imaging, are yet interesting as they potentially also can be applied for brain lesion segmentation.

For MS lesion segmentation Baur et al. (2017) [9] proposed a semi-supervised method, fine-tuning an U-Net using auxiliary manifold embeddings. Another method was proposed by Bai et al. (2017) [6], who showed improved performance by training a CNN for cardiac MR image segmentation in a semi-supervised framework. Furthermore, Perone and Cohen-Adad (2018) [48] extended a mean teacher approach [56] to MRI spinal cord grey matter segmentation and showed increased performance against supervised baseline methods. In computer vision there are several semi-supervised methods proposed for improving results leveraging unlabelled data such as [45, 56, 14] and these would potentially work in medical imaging as well. However, most of the described methods still require a relative large amount of labelled data to perform at full potential.

1.3 Focus of this Work

Recent strives within generative modelling has improved state of the art for various tasks, one being unsupervised semantic segmentation. However, these unsupervised methods still can not compete with supervised methods in terms of performance. We mean to propose a competitive semi-supervised lesion method that leverage unlabelled data and small amounts of labelled data. Our hopes are that we can bridge the gap between supervised and unsupervised learning.

Specifically, this thesis will explore a supervised extension of the method proposed by You et al. for brain lesion segmentation. Particularly, this extension refers to learning the likelihood term in a Bayesian formulation, which today is modelled with TV norm. With a learned prior and likelihood term, we detect lesions similarly as You et al. by casting the segmentation problem as a restoration problem in image space solved by MAP estimation. The goal of this work is to increase the performance of You et al. and compete with supervised methods using only a small amount of labelled data. The focus of this work is to explore supervised ways to learn the likelihood term. Improving the method of You et al. itself and/or the normative prior is not considered focus of this work, since this is already studied in the original paper by You et al.

Firstly, this thesis presents a method, extended from You et al., for binary semantic segmentation of brain lesions using a semi-supervised framework. Secondly, we will investigate how much labelled data is required for achieving comparable performance as unsupervised and supervised baselines models. Since labelled images are expensive, we aim our proposed method to improve results for only extremely limited data, i.e. 1-10 MRI scans.

Brain tumours are a type of lesions and the problem of brain tumour segmentation is clinically relevant and a widely researched problem, and hence a good problem to apply the proposed method on. Additionally, for testing there are relatively big open datasets available from research challenges and also many proposed methods to benchmark against. However, our proposed method can also be trained and used for other semantic segmentation tasks. Especially in a setting where labelled data is scarce but a large amount of unlabelled normal data is available.

Additionally, as a side result of our method, restoring a lesion image to a pseudo ”healthy” counterpart image, could potentially itself be a solution for the pseudo ”healthy” synthesis problem. In clinical practice there is a value in creating a pseudo ”healthy” image for a few reasons. One being segmentation of healthy tissue for follow up analysis of healthy tissue after surgery or radiotherapy. There are many methods for automatic segmentation of healthy brain scans. However, these typically fail when there is a lesion present and if a pseudo ”healthy” image can be found from a lesion image these methods might perform better.

1.4 Thesis Organisation

Firstly, this thesis introduces the problem of brain tumour segmentation, both with clinical background and previous proposed methods. Furthermore, Chapter 2 is devoted to explain related work which our proposed method utilises or benchmark against. Chapter 3 explains our proposed method, that has two different approaches for learning the likelihood. Experimental setup, datasets, visualisation and a quantitative comparison of methods and baselines are provided in Chapter 4. Lastly, the approach, results and future work are discussed in Chapter 5. A full project git repository can be found here: `https://github.com/jkkronk/semisupervised_lesion_seg`.

Chapter 2

Related Work

Our proposed method makes use of previous success within deep learning methods for generative models and semantic segmentation described in this chapter. Furthermore, a semi-supervised baseline method used for performance comparison is described in Section 2.4. Lastly, an overview of related work within pseudo healthy synthesis is given in Section 2.5.

2.1 Variational Autoencoders (VAE)

VAE [33] is a generative model, trained in an unsupervised framework, that efficiently approximates stochastic variational inference and intractable probability density functions of latent variables given higher dimensional training data, X. The model assumes that X originates from unobserved continuous random variables z, which itself is generated from a random process of both the prior distributions $p(z)$ and the likelihood $p(X|z)$. In a probability model perspective, VAE introduces joint probability of X and lower dimensional latent variables z, formulated as marginal likelihood $p(X) = \int p(X|z)p(z)$. To find latent variables z for a given dataset X we need to learn the likelihood $p(z|X)$, i.e. the true posterior density of $p(z|X) = \frac{p(X|z)p(z)}{p(X)}$. The true posterior is most often intractable and impossible to solve and instead of solving it, VAE approximates the true posterior $p(z|X)$ with a probabilistic encoder network $q(z|X)$. Likewise, the likelihood term $p(X|z)$ is approximated with an decoder/reconstruction network. The decoder network samples z from prior learned Gaussian distributions using the reparameterization trick. A vector z is sampled, using learned μ_X and σ_X, with the reparameterization $z = \mu_X + \sigma_X \varepsilon$, $\varepsilon \in N(0,1)$. The reparameterization trick allows the model to be differentiable and backpropagation can be used for training. An overview figure explaining VAE structure can be seen in Figure 2.1.

The encoder and decoder networks is trained using Kullback-Leibler (KL) divergence, a measurement of the divergence between the prior distribution and encoded prior, and the reconstruction loss through the network. Together they form the evidence lower bound (ELBO) which, if maximised, minimises KL divergence and serves as lower bound for $p(x)$. The full loss function of a VAE is formulated in Equation 2.1, where $\hat{X}$ is the reconstructed input X. When applied to imaging problems, normally both the encoding and decoding network is implemented as CNNs. However, they can also be implemented with for example a multiplelayer perceptions network.

$$L = \mathrm{KL}(p(z|X), p(z)) + E_{q(z|X)}(\log p(X|z)) \tag{2.1}$$

Beyond learning the prior distribution $p(X)$, VAEs also makes it possible to sample unseen data from the learned distribution. This makes it possible to synthesise new unseen data, similar to the training data, from

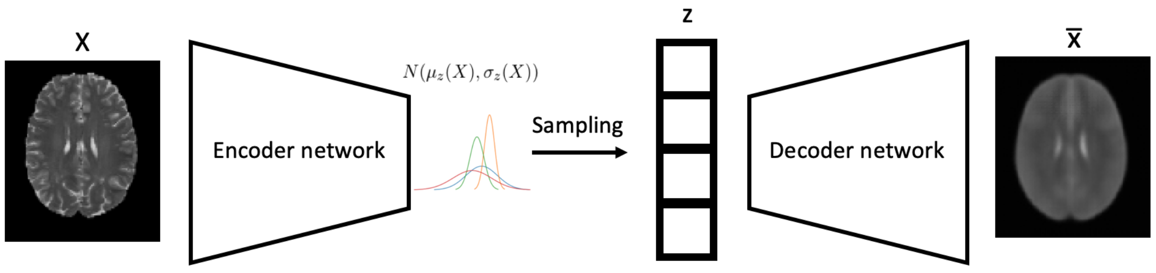


Figure 2.1: Overview structure of VAE.

the learned prior distributions. The synthesis is done by first sampling μ and σ form a normal Gaussian distribution $N(0,1)$. Then a z vector is produced using the the reparameterization trick. Lastly, the z vector is fed into the decoder network, learned to faithfully reconstruct latent variables. Figure 2.2c shows images synthesised from a VAE trained on a training dataset similar to MRI brain scans as seen in Figure 2.2a. Figure 2.2b shows the reconstructed test images from Figure 2.2a. Due to strong and complicated dependencies between pixels, learning the probability density functions for high dimensional data, such as images, is difficult. Nevertheless VAE is a good approximation.

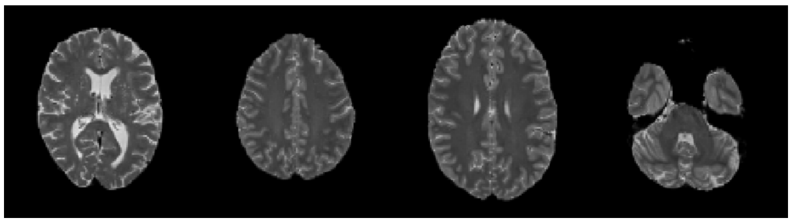
There are several proposed extensions of VAE, such as [21, 49, 34]. These extensions typically aim to better approximate the the true posterior $p(z|X)$. GMVAE, for example, uses Gaussian mixture models, instead of a normal Gaussian priors, to represent latent variables z. This expands the allowed space for latent variables, which leads to better representation of z.

The general goal to learn probability density functions of VAE are similar to the goal of GANs [27]. However, instead of approximating a distribution of latent variables, GANs directly learn the distribution via adversarial networks. GAN has recently become a popular model for image synthesis tasks.

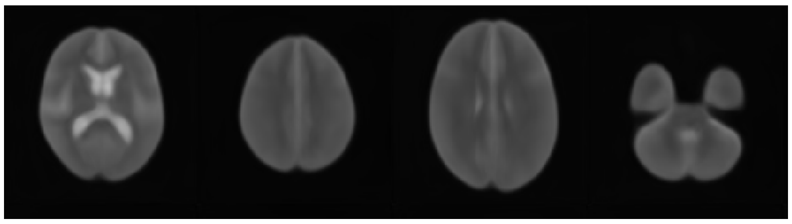
2.2 Unsupervised Lesion Detection via Image Restoration with a Normative Prior

Advances within generative models, like VAE, approximating prior models through latent variables, allowed new methods for unsupervised lesion segmentation by detecting lesions as outliers of distributions from healthy brain scans. New methods, such as [10, 46, 15], used variants of autoencoders or GANs to approximate prior distributions for healthy data. The detection process was then done by reconstructing lesion images and detect lesions as regions with high reconstruction error, i.e. the difference between original image and reconstructed image. The idea was that the trained model can not faithfully reconstruct lesions as they have not been seen in the training data. While achieving competitive results, compared to earlier work, these models produces a considerable number of false positives.

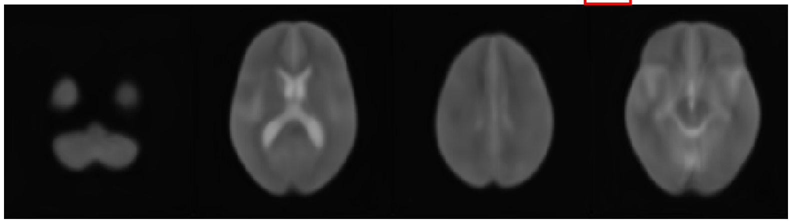
Similarly to these methods, You at al. (2019) [64] propose use of variants of VAE to approximate the prior distributions for healthy brain scans. However, instead of using the reconstruction directly, the detection is cast as restoration in image space. The restoration is solved by MAP estimation, where the prior is combined with a likelihood term. The MAP estimation is solved iteratively with gradient descent to find a restored image where the lesion tissue is replaced with pseudo ”healthy” tissue. The method is inspired by



(a) Examples of test data.



(b) Reconstructed test images from Figure 2.2a.



(c) Random synthesised brain scans by sampling μ_z and σ_z from normal Gaussian distributions, $N(0, 1)$.

Figure 2.2: Results of VAE using axial T2-weighted MRI brain scans from Cam-CAN dataset [54] for training.

Tezcan et al. (2018) [57], who propose a similar method for MR image reconstruction.

With Bayes' theorem, the MAP estimation is formulated as 2.2, where an image with lesion is denoted as $Y \in \mathbb{R}^{\mathrm{N}}$, N is the number of pixels. The restored pseudo "healthy" image is denoted as $X \in \mathbb{R}^{\mathrm{N}}$. $p(Y)$ can be neglected as it does not depend on X. Furthermore, $p(Y|X)$ and $p(X)$ can be separated by taking the logarithm of both sides. Hence, Equation 2.2 is simplified to 2.3. This Bayesian formulation allows to find a restored X, that maximises the posterior distribution $p(X|Y)$, from the likelihood term $p(Y|X)$ and the normative prior $p(X)$. The likelihood term $p(Y|X)$ acts as a data consistency term and punishes changes in the restoration whilst the prior tries to change the image according to a learned prior. You et al. approximated the normative prior using autoencoder models, like VAE and GMVAE trained on healthy data to maximise the $ELBO(X)$. A overview scheme of the method can be seen in Figure 2.3.

$$\underset{X}{\operatorname{argmax}}\, p(X|Y) = \underset{X}{\operatorname{argmax}}[\frac{p(X)p(Y|X)}{p(Y)}] \tag{2.2}$$

$$\underset{X}{\operatorname{argmax}} \log p(X|Y) \propto \underset{X}{\operatorname{argmax}}[\log p(X) + \log p(Y|X)] \tag{2.3}$$

You et al. assumes lesions as structurally compact with sparse gradients and approximate the likelihood $p(Y|X)$ with TV norm [37]. This assumption corresponds to the lesion $D = |X - Y|$ as having a low

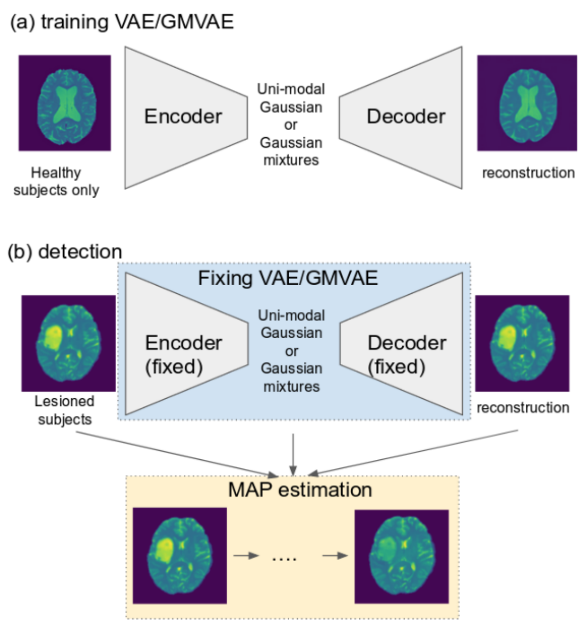


Figure 2.3: Overview of the unsupervised method for lesion detection using a normative prior [64]. Image source from [16].

TV norm With this, a hyper parameter λ is introduced to balance the likelihood and normative prior. By approximating $p(x)$ to $ELBO(X)$ along with approximating $p(Y|X)$ with TV norm, Equation 2.3 is approximated to Equation 2.4.

$$\underset{X}{\operatorname{argmax}} \log p(X|Y) \approx \underset{X}{\operatorname{argmax}}[ELBO(X) + \lambda||X - Y||_{TV}] \tag{2.4}$$

As both the $ELBO(X)$ and the TV norm are differentiable, with respect to the input image X, the MAP estimation in Equation 2.4, is solved by gradient descent. In each iterative step, X_i is updated with Equation 2.5, where $X_0 = Y$ and $X_n = \hat{X}$ (the restored image). The gradient descent steps, with step size α, continues until the function has converges, i.e. gradients of the $ELBO(X)$ and the TV norm are close to zero for a number of iterations.

$$X_{i+1} = X_i + \alpha \frac{\partial}{\partial X_i}(ELBO(X_i) + \lambda||X_i - Y||_{TV}) \tag{2.5}$$

Determining the hyperparameter λ is done empirically by observing restoration of healthy subjects. Restoring healthy subject should yield a restoration error as small as possible. By choosing a high λ, whilst still only little or no restoration error is seen when restoring healthy data, should result in good performance when restoring lesions. You et al. showed that λ was quite robust and good results was still found with a λ slightly off.

As described earlier, detecting the lesion D is done by taking the absolute difference of the original and restored image, as $D = |X - \hat{X}|$. This results in a lesion map with continues values where pixel values in D are $0 \leq D \leq 1$. To create a semantic binary segmentation from the lesion map, a threshold must be chosen.

You et al. propose to choose the threshold by first restoring healthy images, which should result in a lesion free segmentation. Secondly, the threshold is set as the lowest threshold to satisfy a pre set false positive rate for the healthy segmentations. This method was first described by Konukoglu and Glocker (2017) [35].

You et al. outperform previous state of the art results for unsupervised lesion detection. Tested on the BraTS dataset for tumour segmentation, the proposed method achieves up to AUC = 0.83 and dice score up to $0.46(\pm 0.23)$.

2.3 U-Net

Ronneberger et al. (2015) [51] propose a CNN architecture, named U-Net, for fast and accurate biomedical image segmentation in a supervised framework. With a deep convolutional network, based on work from Long and Selhammer [38], Ronneberger et al. created a state of the art image segmentation network. Still today, methods using variants of U-Net achieve competitive results for various medical image segmentation tasks [29]. Ronnerberger et al. also showed that U-Net can achieve good performance even for relatively small datasets, by applying data augmentation and keeping the segmentation task relatively domain specific.

The network architecture of U-Net, seen in Figure 2.4, has a symmetric encoder-decoder structure forming an U shaped structure, hence the name. The encoder contracts the input image and consists of four layers where in each layer two 3x3 convolution operators, followed by a rectified linear unit and a 2x2 max pooling operator are applied. The max pool operator down samples the image by two, while each layer also increases feature layers by two. The decoder network reconstructs the segmentation by upsampling the features in the bottleneck to the same shape as the input image. In each of the four decoder layers the input is first upsampled and then concatenated with the corresponding encoder layer in order to localise the features. After the concatenation, similarly as in the encoder, two 3x3 convolution operators followed by a rectified linear unit are applied. Lastly, a convolution operator layer is used to get the right number of output classes, which in binary classification is one. The network is normally trained by maximising the similarity between the output and a ground truth segmentation map.

In the original paper, Ronnerberger's propose the architecture for segmentation of neuronal structures in electron microscopic stacks with input size of 512x512. The bottleneck has a size of 28x28, with a receptive field of 78x78 before up sampling. In total, the original U-Net architecture has 7,759,521 trainable parameters.

Dong et al. [22] applied a U-Net architecture on brain tumour segmentation task participating in BraTS challenge 2015. A depth of 4 max pool layers and an input size of 240x240 2d slices was used. This achieved a Dice score for the whole tumour of 0.86. Additionally, 3d U-Net [17], using 3d convolutional operators, have been proposed. A 3d U-Net for brain tumour segmentation leverage the volumetric 3d nature of MRI. However, these models require lots of training data and are usually expensive to train.

2.4 SimCLR - Contrastive Learning

Lately, methods for semi-supervised learning, leveraging unlabelled data, have become popular due to their ability to substantially improve performance in various tasks. [14, 56, 45] are just a few of many examples where using unlabelled data improved performance for supervised models. SimCLR [14] is a contrastive self-supervised learning algorithm that showed improved state of the art performance with a semi-supervised task. SimCLR aims to learn good representations from unlabelled data using strong data augmentation and contrastive loss. The method can be used independent of architecture, in problems where for example labelled data is scarce but a large dataset of unlabelled data is available.

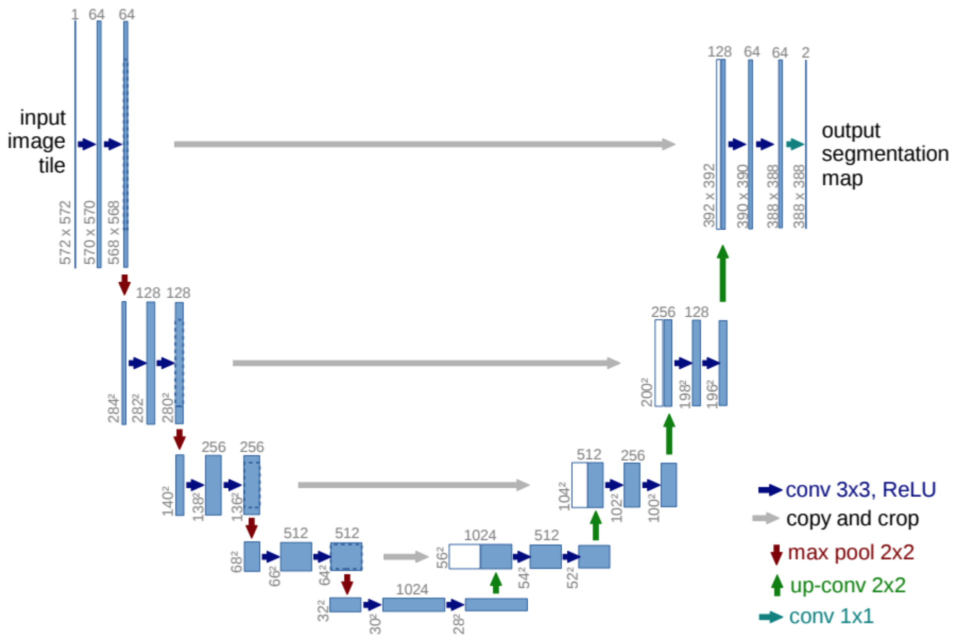


Figure 2.4: Network architecture for the original U-Net. Image source from [51].

The general goal of contrastive learning is to learn representations by punishing differences, maximising agreement, between two similar inputs and contrary contribute to differences, minimising agreement, between two dissimilar inputs. This is done with Noise Contrastive Estimator loss function in Equation 2.6. Here, x^+ is an augmented version of x, whereas x^- is a dissimilar data from x. The $sim()$ function is a distance function, proposed to be cosine similarity or dot product, and $g()$ is the neural network with trainable weights. An overview of the training scheme of simCLR can be seen in Figure 2.5.

$$NCE_{loss} = -\log \frac{exp(sim(g(x), g(x^+)))}{exp(sim(g(x), g(x^+))) + \sum_{k=1}^{K} exp(sim(g(x), g(x_k^-)))} \tag{2.6}$$

SimCLR can be applied to a wide variety of neural networks using a semi-supervised framework. For example, the U-Net segmentation network might improve its performance by first pretraining the encoder with constrictive loss using unlabelled data. The encoder is then learned to maximise agreement for similar inputs, and correspondingly minimise agreement for different inputs. Following, the whole U-Net is then fine-tuned with labelled data. The pretraining potentially helps learning important features in the encoder and initialises parameters better for fine-tuning on labelled data. This idea is similar to how transfer learning would improve performance.

2.5 Pseudo Healthy Synthesis

The problem of creating a pseudo ”healthy” image from a lesion image is, due to the nature of the problem, a challenging topic within medical imaging. Firstly, there is a lack of paired non healthy and healthy data,

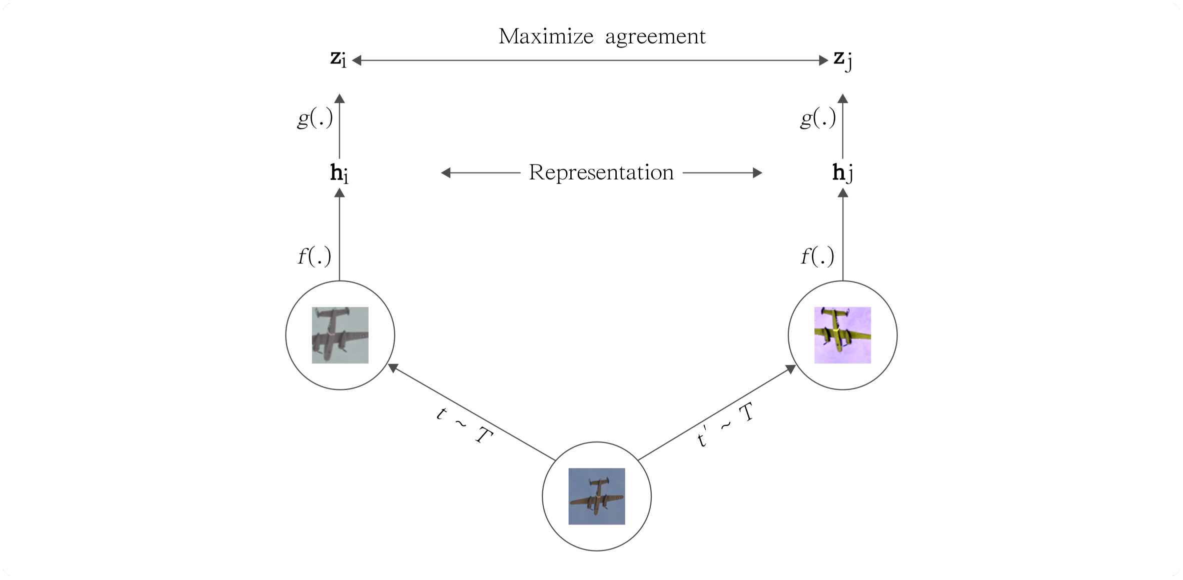


Figure 2.5: Overview of training scheme for contrastive loss using simCLR [14]. Image source from [14].

as these data pairs is very hard or impossible to collect. Secondly, coming up with good quantitative metrics for evaluation of pseudo ”healthy” images are hard. Lastly, creating pseudo ”healthy” images, is a ”one to many problem” and a lesion image might have many possible solutions. Nevertheless, a few methods for synthesising pseudo ”healthy” images has been proposed and lately these methods has become so good that, for an untrained eye, its hard to distinguish a synthesised image to a real one.

Bowles et al. (2017) [11] propose a segmentation model based on image restoration where pairs of T1 and FLAIR MRI sequences were used to find a pseudo ”healthy” counterpart in the FLAIR sequence. Even if this work is promising, as it relies that the lesions is visible in FLAIR but not in T1 sequence, its practical use is limited. Since brain tumours are more or less visible in all sequences this approach is not suited for our problem.

Xia et al. (2020) [63] propose an adversarial learning method to synthesise pseudo ”healthy” brain images from lesion images. The approach is to first disentangle the lesion image to a corresponding pseudo ”healthy” image and a lesion segmentation map. This is done with a generator and segmentation network. Next, the two disentangled components get reconstructed back to the original image via a reconstructor network. Although the main goal from this work is not a lesion segmentation, the authors provided segmentation results from both the output of the segmentation network and through the reconstruction error. When trained in an unpaired fashion, with no paired segmentation lesion pairs, and evaluated on BraTS dataset for tumour segmentation, the method achieves a Dice score of 0.74 for the segmentation output and 0.7 through the reconstruction error. However, as this method requires ground truth segmentation maps, (unpaired or paired), it is not suitable for a setting, like ours, with few ground truth annotations. Results of synthesised lesion images can be seen in Figure 2.6. One can see that tumour areas gets reconstructed faithfully, however the method also changes the healthy tissue, which is not desirable.

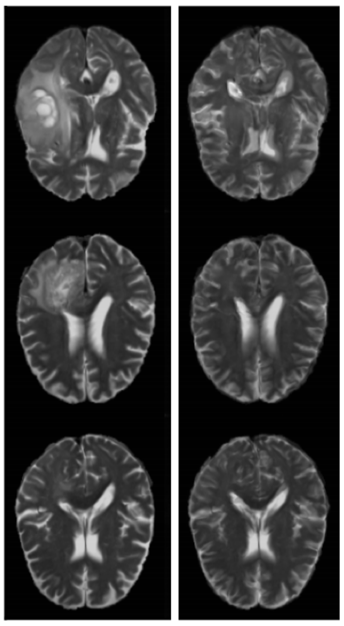


Figure 2.6: Resulting synthesis of lesion images. Left: Original lesion images; Right: Synthesised pseudo ”healthy” images. Image source from [63].

Chapter 3

Methods

In this chapter we are first going to present an overview of the proposed methods used for brain lesion detection, resulting in a binary segmentation, via image restoration. Secondly, learning the normative prior is explained in Section 3.2. Following, two proposed approaches of learning the likelihood is described in Section 3.3. Lastly, Section 3.7 describes used data augmentation for our method.

3.1 Overview

Our proposed method can be seen as a supervised extension of the method proposed by You et al. [64] for unsupervised lesion detection via image restoration with a normative prior. Our idea is to leverage the prior information, learned in an unsupervised framework, combined with a likelihood term, learned in a supervised framework with limited amounts of data, to restore a lesion image into a pseudo "healthy" image. The method detects lesions and results in a binary semantic segmentation, tested on whole tumour segmentation problem.

In this work we assume that a lesion can be modelled as additive noise, $D \in \mathbb{R}^{\mathrm{N}}$, to a pseudo "normal" image, $X \in \mathbb{R}^{\mathrm{N}}$. An image with a lesion is then described as $Y = X + D$. If X is available, the lesion can be obtained by the absolute difference between Y and X, $D = |Y - X|$. Since X is not given, first we need to restore X from Y. This restoration corresponds to the pseudo "healthy" synthesis task, which we solve using MAP estimation and Bayes theorem, similar to You et al., as in Equation 2.2 and 2.3. Here, $p(Y|X)$ is the likelihood term and $p(X)$ is the normative prior. The MAP estimation optimisation problem is solved through gradient descent, where X is iteratively updated with the gradient of $\log p(Y|X) + \log p(X)$, with respect to X, as Equation 3.1. α is the step size of the gradient descent and initially $X_0 = Y$. Restoration continues n steps until convergence.

$$\begin{aligned} X_{i+1} &= X_i - \alpha \left[\frac{\partial}{\partial X_i} [\log p(X_i) + \log p(Y|X_i)] \right] \\ &= X_i - \alpha \left[\frac{\partial}{\partial X_i} \log p(X_i) + \frac{\partial}{\partial X_i} \log p(Y|X_i) \right] \end{aligned} \tag{3.1}$$

Before carrying on with more specific details about the normative prior and the likelihood, let's recall that You et al. divides their method into two steps: 1) learning a normative prior of a healthy brain autonomy and 2) casting lesion detection as image restoration and detecting lesions as regions with high restoration error. Likewise, we divide our method into these two steps, where the two steps are identical in both methods. However, our extension includes learning the likelihood term, which in an unsupervised framework is not possible. Hence we add an extra step in our method. The full proposed method can then be divided into the following three steps:

1. Learning a normative prior using unsupervised learning
2. Learning the likelihood term using supervised learning
3. Restore lesion regions to pseudo ”healthy” and detect lesions as regions with high restoration error

3.2 Learning the Prior

We make use of the same method for learning prior distributions for normal data as You et al. In our case the normal data corresponds to a healthy brain autonomy. Since learning the prior is not the focus of this work, we refer to the work of You et al. for a more in depth description and further reading.

As described in Section 2.1, VAEs can be used to approximate probability density functions of latent variables $z \in \mathbb{R}^n$ given a higher dimension input $X \in \mathbb{R}^N$, where $n << N$. VAEs define a lower bound for approximating the prior distribution $p(X)$, using the reconstruction loss and KL divergence, together called $ELBO(X)$. While training, we optimize parameters of VAE network to maximise $p(X)$ by minimising the $ELBO(X)$ for a healthy dataset. While testing, minimising the $ELBO(X)$ for abnormal test data results in a pseudo ”normal” counterpart image given the learned prior.

The network architecture for the used VAE is, identical to what You et al. propose, a CNN with residual blocks. The encoder's residual blocks, shown in Figure 3.1, contain a combined straight path and shortcut path. The straight path contains two consecutive convolutional operators (3x3), with stride 1 and stride 2. Each convolution has a leaky rectified unit as activation function. The shortcut consists of one convolution (3x3), with stride 2. Similarly, the shortcut also has a leaky rectified unit as activation function. Following each convolution is a batch normalisation operator that allows for higher learning rates and ease training. The encoder is built up of six residual blocks, where an input of 128x128 results in latent variables of size 512. The decoder consists, similar to the encoder, of six up-convolutional residual blocks. These up-convolution blocks are identical as the encoder residual blocks but use up-convolution operators, instead of normal convolution operators, to decode the image from the latent variables. The 512 sampled latent variables result, after the decoder, in a 128x128 output. For further details, see architecture printout in Appendix Section A.3.

For our proposed method any autoencoder model, for example VAE and GMVAE, approximating normative prior distribution $p(X)$ can be used. However, since the focus of this work is not to extend, improve or explore different models approximating the normative prior, a VAE network is used. VAE is a well known and an accepted model to approximate the prior.

In MAP estimation, the normative prior $p(X)$ is restricted with a likelihood term $p(Y|X)$, punishing large deviations from Y. In our case, we use the likelihood term to restrict $ELBO(X)$ and punish changes in healthy tissue.

3.3 Learning the Likelihood

Our approach is to use, either an implicitly or explicitly, trained segmentation network, $NN_\theta(Y, X)$, that multiplied by $-ELBO(X)$ approximates the gradient of likelihood term $p(Y|X)$ and acts as the data con-

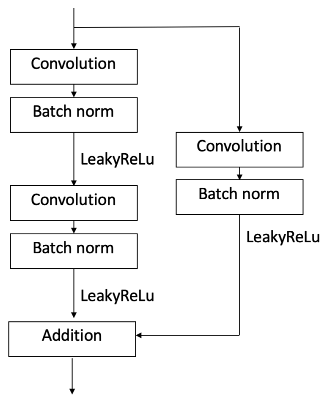


Figure 3.1: Residual block of encoder as implemented in the VAE.

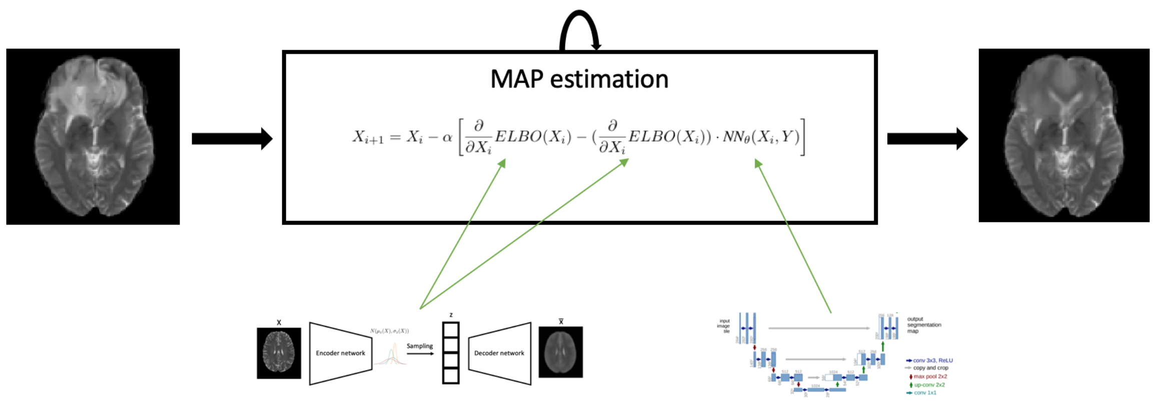

MAP estimation
$X_{i+1} = X_i - \alpha \left[\frac{\partial}{\partial X_i} ELBO(X_i) - (\frac{\partial}{\partial X_i} ELBO(X_i)) \cdot NN_\theta(X_i, Y) \right]$
Encoder network
Sampling
Decoder network
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

Figure 3.2: Schematic overview of the restoration process.

sistency term for the normative prior $p(X)$. The idea is that the prior restores lesions good enough, but fails with encapsulating detailed variances which result in normal tissue being wrongly restored. The proposed segmentation network restricts normal tissue from being restored by the normative prior. The gradient descent step in Equation 3.1, can with our approach be written as Equation 3.2 where $p(X)$ is approximated with $ELBO(X)$. The segmentation network is a neural network, with sigmoid output activation function, that gives a pixel wise probability, $0 \leq p \leq 1$, for lesions. If approximated correctly, normal tissue should result in the segmentation network to output one and hence cancel out gradients of $ELBO(X)$. Respectively, lesion tissue should result in zero and hence allowing $ELBO(X)$ to change these regions. A schematic overview of the restoration process is illustrated in 3.2.

$$X_{i+1} = X_i - \alpha \left[\frac{\partial}{\partial X_i} ELBO(X_i) - (\frac{\partial}{\partial X_i} ELBO(X_i)) \cdot NN_\theta(X_i, Y) \right] \tag{3.2}$$

While testing, we make use of both the separately trained prior and likelihood term to obtain a pseudo ”healthy” image, i.e. in our case the tumour regions are restored to ”healthy” while healthy regions are not. The segmentation map is then obtained by the difference between the restored and original image, $S = |X_n - Y|$. Algorithm 1 illustrates an overview of how the restoration is performed while testing.

Algorithm 1 Testing: $f : X \to \hat{S}, X \in \mathbb{R}^{\mathrm{N}}$

```
$X_0 = Y$
**for** i in range($n$) **do**
    $\hat{X}_s = \mathrm{NN}_\theta(Y, X_i)$
    $X_{i+1} = X_i - \alpha \left[\frac{\partial}{\partial X_i} ELBO(X_i) - (\frac{\partial}{\partial X_i} ELBO(X_i)) \cdot \hat{X}_s\right]$
**end for**
$\hat{S} = |X_n - Y|$
```

In Algorithm 1 there are two parameters to set, α and n. α is the size of the gradient descent step and should practically be set as low as possible, keeping in mind that a low step size increases convergence time for the optimisation problem. The restoration iteratively continues n steps, until convergence, i.e. when the gradients are close to zero for a number of steps.

The proposed network architecture for the segmentation network is a modified U-Net [51] where the network is reduced considerably in order to prevent overfit. An original U-Net architecture has proven to perform good on limited amount of data, however in our setting we have extremely limited amount of data. Additionally, the final segmentation result is a combination of the segmentation network and normative prior output, hence the segmentation network does not need to perform as if it is a standalone segmentation network. Reducing the network size and the space allowed for network parameters should help reduce overfit. We propose the same encoder and decoder with skip connections as the original U-Net. Nonetheless, we reduce layers to 3 and initial features to 2. This shrinks the U-Net substantially, whilst hopefully producing a better end result since overfit is potentially reduced. We name our network as a ”shallow U-Net”. The detailed network architecture printout can be found in Appendix Section A.2.

We train the segmentation network with Adam optimiser [32], an optimisation method proven to successfully train deep neural networks efficiently. Adam is an optimisation algorithm that is based on, AdaGrad [23] and RMSProp [59], with adaptive estimation of lower order momentum. To prevent overfit we perform early stop when the validation loss starts to increase. Learning rate is set with a naive approach, where the network is trained and learning rate is chosen to the best, in terms of performance for the validation data. We propose two different ways of learning the likelihood: one implicit and one explicit way with different loss functions.

3.3.1 Implicit Learning

The first proposed approach to train the segmentation network is in an iterative implicit manner, optimised with Dice loss function as Equation 3.3. Here, the segmentation network is trained to find a good implicit segmentation in each restoration step. $Dice_{Loss}$, defined as Equation 3.4 and also known as F1-Loss, have been proven to work good for training CNNs for medical images [41]. Especially, Dice loss works good for problems with unbalanced classes such as brain tumour detection. Other loss functions, binary cross entropy and structural similarity, were also tested but Dice loss seemed to fit our problem best in terms of performance and training speed. $S \in \{0, 1\}$ corresponds the ground truth semantic segmentation, labelling lesions as 1 and normal tissue as 0. Training the network in this implicit manner results in a segmentation that will punish deviations outside of the lesions while restoring.

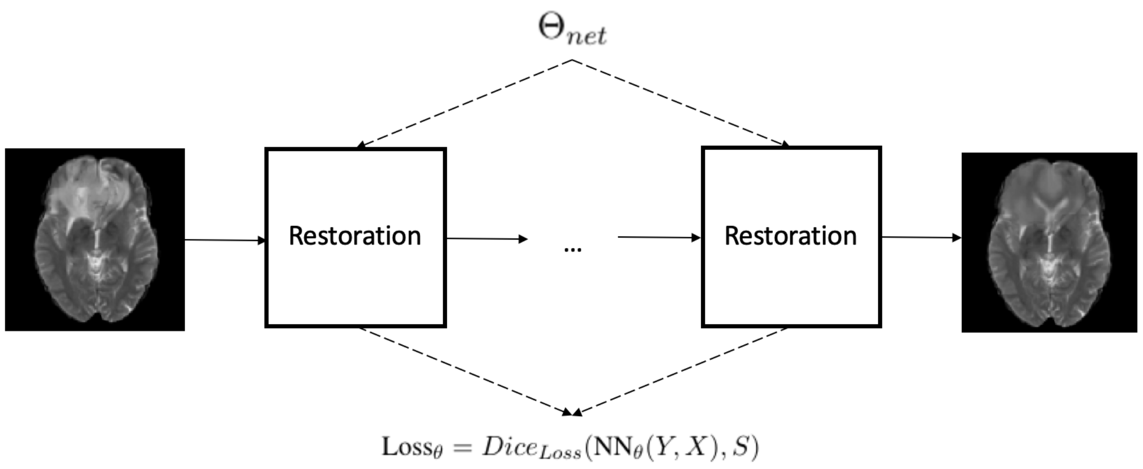

Θ_{net}
Restoration
...
Restoration
$\text{Loss}_\theta = Dice_{Loss}(\text{NN}_\theta(Y, X), S)$

Figure 3.3: Training scheme for the neural network $\text{NN}_\theta(Y, X)$, where each restoration block corresponds to Equation 3.2. The dotted arrows show the graph for backpropagation and the continues arrows show the restoration process.

$$\text{Loss}_\theta = Dice_{Loss}(\text{NN}_\theta(Y, X_i), (1 - S)) \tag{3.3}$$

$$Dice_{Loss}(X, \hat{X}) = 1 - \frac{2X\hat{X} + 1}{X + \hat{X} + 1} \quad 0 \leq X \leq 1, \hat{X} \in \{0, 1\} \tag{3.4}$$

Training the network is done in an iterative manner, as Figure 3.3, which correspond to restoring and training the image simultaneously as Equation 3.5. Here, n is the number of restoration steps and m is the number of epochs. This iterative approach allows the network to see every restored image which, in theory, extends the dataset to include lower grade tumors restored from higher grade tumours. One could see this as a sort of data augmentation. Additionally, this iterative approach allows data augmentation, further explained in Section 3.5, to be performed inside each restoration step. This is implemented by having two forward passes through the network, one forward pass with augmented input for the loss function and one forward pass with direct input for the next restoration step. Two forward passes allows each restoration step to be more faithful since there is no augmentation performed for the step, only on the loss function. Additionally, a more aggressive data augmentation can be used since the normative prior does not need to reconstruct the augmented image. For better and more stable optimisation of network parameters θ, the loss is aggregated and θ updated after each restoration. The full algorithm for the training is illustrated in Algorithm 2.

$$\theta = \underset{\theta}{\text{argmax}} \sum_{j=0}^{m} \sum_{i=0}^{n} \text{Loss}_\theta \tag{3.5}$$

3.3.2 Explicit Learning

The second proposed approach, for training the segmentation network, is in an explicit manner, where the network is optimised to achieve the best final segmentation, rather than, as the implicit approach, optimised

Algorithm 2 Training Implicit: $\theta = \text{argmax}_\theta \sum_{j=0}^{m} \sum_{i=0}^{n} \text{Loss}_\theta$

```
for j in range(m) do
    $X_0 = Y$
    $L = 0$
    for i in range(n) do
        $\hat{X}_s = \text{NN}_\theta(Y, X_i)$
        $L = L + Dice_{Loss}(\hat{X}_s, (1 - S))$
        $X_{i+1} = X_i - \alpha \left[\frac{\partial}{\partial X_i} ELBO(X_i) - (\frac{\partial}{\partial X_i} ELBO(X_i)) \cdot \hat{X}_s\right]$
    end for
    $\Theta_{net} = \Theta_{net} - \lambda_1 \frac{\partial}{\partial \theta_{net}} L$
end for
```

to achieve a good gradient step to a good pseudo ”healthy” restoration. The explicit learning approach has the same restoration fashion and network architecture as the implicit approach. However, training aims to minimise the loss function as Equation 3.6, where the network learns to optimise the final segmentation. X_i is the partly restored image in each restoration step and $X_0 = Y$ is the original input image. The ground truth segmentation S is defined as 0 for healthy tissue and 1 for lesion tissue. K is an activation function factor, explained in the next paragraph. An overview of the training is further described in Algorithm 3. This approach is promising since it is aimed to optimise the final segmentation and the segmentation network itself is guiding the prior term, rather than restricting it as in the implicit approach.

$$\text{Loss}_\theta = Dice_{Loss}(\tanh(K(X_i - X_0)^2), S) \tag{3.6}$$

Algorithm 3 Training Explicit: $\theta = \text{argmax}_\theta \sum_{j=0}^{m} \sum_{i=0}^{n} \text{Loss}_\theta$

```
for j in range(m) do
    $X_0 = Y$
    $L = 0$
    for i in range(n) do
        $\hat{X}_s = \text{NN}_\theta(Y, X_i)$
        $X_{i+1} = X_i - \alpha \left[\frac{\partial}{\partial X_i} ELBO(X_i) - (\frac{\partial}{\partial X_i} ELBO(X_i)) \cdot \hat{X}_s\right]$
        $L = L + Dice_{Loss}(\tanh(K(X_{i+1} - X_0)^2), S)$
    end for
    $\Theta_{net} = \Theta_{net} - \lambda_1 \frac{\partial}{\partial \theta_{net}} L$
end for
```

Since the segmentation map, $(X_i - X_0)$, has continues values in the interval $-1 \leq X_n - X_0 \leq 1$ it can not be directly compared and optimised to the binary ground truth segmentation. Hence, an activation function must map values of the segmentation map to a binary map. This could be done by a customised hyperbolic tangent function, as $\tanh(x^2) = \frac{e^{x^2} - e^{-x^2}}{e^{x^2} + e^{-x^2}}$. This activation function is shown in Figure 3.4. The segmentation map power by 2 allows both positive and negative changes to be labelled as lesions. The problem with the proposed activation function is that it does not allow small values in the segmentation map to be mapped to 1. Since our data is normalised between 0 and 1, intensity value for the segmentation map will naturally only be small values. To solve this, a K hyper parameter is added as a factor to the

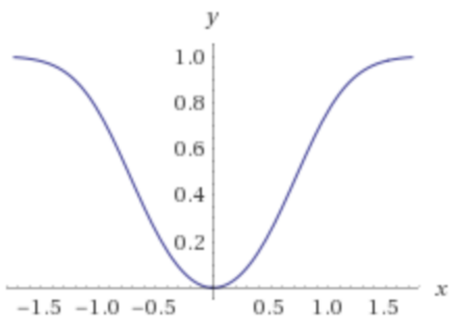


Figure 3.4: Plot of activation function: $y = \tanh(x^2)$.

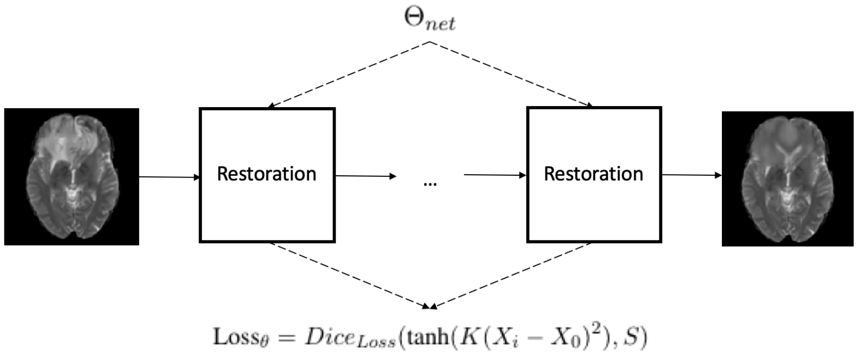

Θ_{net}
Restoration
...
Restoration
$\text{Loss}_\theta = Dice_{Loss}(\tanh(K(X_i - X_0)^2), S)$

Figure 3.5: Training scheme for the neural network $\text{NN}_\theta(Y, X)$, where each restoration block corresponds to Equation 3.2. The dotted arrows show the graph for backpropagation and the continues arrows show the restoration process.

segmentation map. Choosing a correct value K turns out not to be trivial. One could say that any false positives in the segmentation map should be punished and hence set K to a large value. However, setting a very large value of K affects the gradients of network parameters θ while training and might result in unstable learning. We set the K parameter empirically with the help of an additional validation set and choose a K that maximises the performance for both training and validation set.

In difference to the implicit approach, the explicit approach can be trained in two different fashions. Either, backpropagation is done in each detached restoration step and losses aggregated, shown in 3.5, similar to how the implicit method is trained. Else, the restoration is kept connected and backpropagation of loss is only performed once after the restoration is complete. This fashion, shown in 3.6, has similarities to how recurrent neural networks are trained. This training fashion requires more memory which can be solved by lowering the batch size. It is not straight forward which approach is theoretically best for our setting. However, backpropagation from the end result is somewhat more appealing since it optimises only the end segmentation result. To keep the implementation simple and be able to easily compare the implicit and explicit approach, we choose to implement the explicit approach with a disconnected graph and aggregated loss, i.e. the same implementation as the implicit approach.

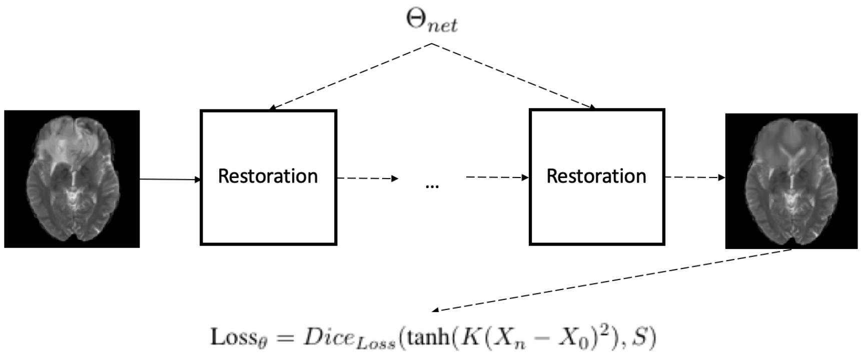

Θ_{net}
Restoration
...
Restoration
$\text{Loss}_\theta = Dice_{Loss}(\tanh(K(X_n - X_0)^2), S)$

Figure 3.6: Training scheme for the neural network $\text{NN}_\theta(Y, X)$, where each restoration block corresponds to Equation 3.2. The dotted arrows show the graph for backpropagation and restoration process.

3.3.3 Gradient Learning

While exploring different approaches for learning the likelihood term we tried several different strategies. Although the restoration process is similar in all of the tested approaches, there are many ways to define a loss function for training the network.

One other approach we tried was to learn from the gradient descent steps. The idea was drawn from the work of Adler and Öktem (2017) [1] who proposed a method for solving ill-posed inverse problems using iterative deep learning networks. Their approach consists of learning gradient steps to solve a optimisation problem similar to our goal. Our idea was to update the gradient step with Equation 3.7 where the neural network guides the gradient in a learned direction. The network was trained in an iterative manner while restoring and the loss function was defined to minimise the difference between the gradient and the segmentation map as in Equation 3.8. However, this loss function is only valid for the first iteration of the restoration since there is only a ground truth segmentation for the full lesion available and naturally in each restoration step the lesion is partly restored. Hence, when the image is partially restored the segmentation map is not valid and the network will train to restore already restored "healthy" regions. This approach is promising but the drawback of only being able to learn from one restoration step limits its performance.

$$X_{i+1} = X_i - \alpha \left[\frac{\partial}{\partial X_i} ELBO(X_i) - \text{NN}_\theta(Y, X) \right] \tag{3.7}$$

$$\text{Loss}_\theta = Dice_{Loss}\left(\left[\frac{\partial}{\partial X_i} ELBO(X_i) - \text{NN}_\theta(Y, X) \right], S \right) \tag{3.8}$$

3.4 Choosing Threshold for Binary Segmentation

The restoration error found when restoring image X to $\hat{X}$ is the segmentation map, $\hat{S} = |X - \hat{X}|$. Here, the segmentation map has continues values in range $0 \leq \hat{S} \leq 1$. To create a binary segmentation map, as $S \in 0, 1$, we need to choose a threshold for the continues segmentation map. With the available data there are two proposed options to achieve this.

First option is a naive approach of simply choosing the best threshold that yields best performance while restoring the training data. The best threshold is considered to be the threshold with the highest Dice score, calculated as in Equation 3.9. This approach assumes that the training dataset generalises well outside the training data. However, in our setting, since we have a large dataset of unlabelled data, we can also leverage this data to set a good threshold.

$$\text{Dice} = \frac{2||X \cap X_n||}{|X| + |X_n|} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{3.9}$$

The second option for calculating threshold, also the approach of You et al, is to leverage the unlabelled data. In our setting the unlabelled data is consisting of ”healthy” image scans without lesions. If this data is being restored, the resulting segmentation map should not show any lesion regions. This approach was first introduced by Konokoglu and Glocker (2017) for reconstructing subject-specific effect maps. The idea is to first choose an acceptable false positive rate for the healthy brain images. After restoring these and obtaining the segmentation map, the threshold is chosen to be as low as possible, whilst achieving the set false positive rate. This approach allows for pre-defined amounts of change in healthy tissue and assumes that there will be more change in regions with lesions.

In experiments we will report results of both these approaches to investigate which is best for our setting. Even if the second option is good in theory, the training set might generalise the test good enough and result in higher performance, especially when the training set is large.

3.5 Data Augmentation

To improve generalisation capabilities of the used neural networks, we make use of data augmentation. Recall that our problem only allows use of a limited labelled training dataset. Data augmentation plays an important role in ”increasing” our dataset, which leads to improved generalisation of the used neural networks. Better generalisation leads to less overfit and better performance outside the training data. We have implemented our data augmentation as a composition of random transforms and augmentation is done on the fly. The proposed transforms was chosen after reading the review paper of Nalepa et al. (2019) [43].

We used two different sets of transformations while training our methods. First data augmentation, as seen in Table 3.1, was used to augment data for training the VAE. For each new training epoch pre-processed data gets randomly transformed, which results in a ”infinite” dataset.

The second set of transformations, as seen in Table 3.2, is performed, in each restoration step, while training the segmentation network with the implicit and explicit approach. By performing two forward passes through the network, one forward pass for the loss function and the second forward pass to take a gradient step, a more aggressive data augmentation can be used. Since the VAE will only see non augmented data it can still achieve a faithful prior, whilst the segmentation network leverage a more aggressive data augmentation. Figure 3.2 shows random data augmentation for a single slice using data augmentation as Table 3.2. Data augmentation was implemented using imgaug library, `https://imgaug.readthedocs.io/en/latest/`.

Transform	**Parameter**
Horisontal Flip	0.5%
Rotate	random(± 10 deg)
Elastic Deformation	$\alpha =$ random$(0, 100)$, $\sigma = 10$
Scale	random($\pm 0.20\%$)
Average Blur	random(0,4)
Linear Contrast	random(± 0.2)
Intensity Multiply	random(± 0.2)

Table 3.1: Random transformations used for data augmentation while training the **VAE**.

Transform	**Parameter**
Horisontal Flip	0.5%
Vertical Flip	0.5%
Rotate	random(± 20 deg)
Elastic Deformation	$\alpha =$ random$(0, 200)$, $\sigma = 20$
Scale	random($\pm 0.20\%$)
Average Blur	random(0,4)
Linear Contrast	random(± 0.3)
Intensity Multiply	random(± 0.2)

Table 3.2: Random transformations used for data augmentation while training the **segmentation network**.

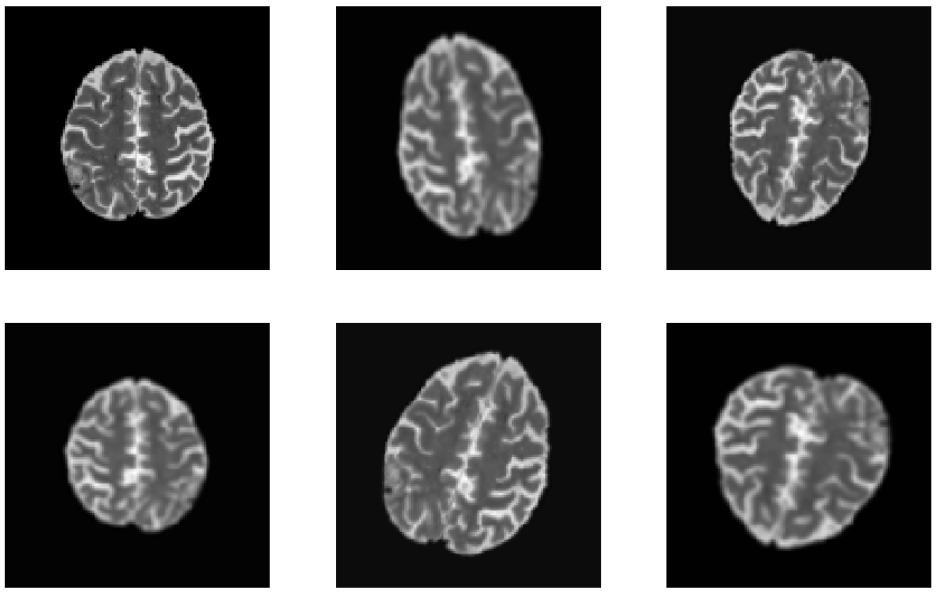


Figure 3.7: Data augmentations with random transformations as Table 3.2 on same axial T2-weighted slice. The top left image is the original non augmented input image.

Chapter 4

Experiments and Results

This chapter will explain the experimental setup and present results from our method and baselines. Firstly, after description of experimental setup, we will visualise results for each baseline and the proposed methods separately. Secondly, a quantitative comparison between the methods will be presented.

4.1 Experimental Setup

4.1.1 Datasets

Our method was trained and tested on brain tumour segmentation problem, using T2-weighted 2d axial slices. T2-weighted sequence was chosen because of its extensive use for both healthy and brain tumour MRI scans. It is possible for the method to be extended and applied leveraging 3d volumetric inputs and more channels, e.g. T1-weighted, T1-weighted with contrast enhancement and FLAIR. However, extending the method to 3d volume segmentation will introduce more spatial dependencies which might require a larger dataset. Additionally, 3d volumetric segmentation can be seen as out of scope for this work and hence we did not explore it further. Extending the method to include more sequences might be possible, however this would make implementation of restoration process more complicated as every sequence needs to be restored separately and lesion detection might not be as straight forward. As the focus of this work was to present proof of a working method, extra sequences were not considered essential or important and were therefore left out.

Two datasets, Cam-CAN [54] dataset and BraTS 2017 challenge dataset [7], were used for training our neural networks. Moreover, a hold out subset of the BraTS challenge dataset was used for testing. The Cam-CAN dataset consists of 653 T2-weighted MRI scans of healthy adult individuals. The 653 subjects were randomly divided into 500 training subjects and 153 validation subjects. The scans were already skull-stripped and co-registered to the same anatomical template. The T2-weighted scans are 256x256x192 and with resolution of 1x1x1 mm per voxel. We use the Cam-CAN dataset as both an unlabelled dataset for training the VAE, and also for choosing the threshold to achieve a binary segmentation map.

The BraTS 2017 challenge training dataset consists of 285 MRI scans of glioblastoma (HGG) and lower grade glioma (LGG) patients with ground truth segmentations. The dataset is obtained from several different institutions and machines, which results in relatively high variances in sizes, resolution, sharpness and contrast within the dataset. Ground truth segmentations were manually done by several raters following a standard protocol, and approved by neuroradiologists. Annotations of tumours consist of four labels, GD-enhancing tumour, peritumoral edema, necrotic and non-enhancing tumor. However, because of the nature of our approach, we labelled all of these mentioned labels as tumour tissue and our segmenting problem is what BraTS challenge calls whole-tumour segmentation. The BraTS dataset was already skull-stripped, co-registered and interpolated to the same resolution. Since the official test dataset of BraTS challenge dataset is not fully available for public, we randomly separated the training set into a new training set of 235 subjects and a test set of 50 subjects.

To understand how much labelled data is required for our method, we trained each model with 1/3/10/30/100 number of subjects. To understand the variation between models trained on different subjects, each model, with 30 or less subjects, was trained 5 separate times with randomly chosen subjects. The validation set was randomly chosen as 15% of the total slices from the chosen subjects. To be able to compare results, identical subjects were trained on the proposed and baseline methods.

4.1.2 Data Preprocessing

Our data preprocessing consists of steps as described in the list below. All steps are further explained in this subsection.

1. Bias field correction
2. Normalising (0,1)
3. Resize and Crop to 128x128
4. Histogram matching

Data preprocessing is important when working with MRI mainly because of two reasons. Firstly, in raw MRI there is often a bias field present which is expressed as an added low-frequency and smooth signal to the underlying real data. The bias field is causing pixel intensities of the same tissue type to be dependent on where they are spatially located. Bias field removal is a well studied topic. N4ITK, proposed by Tustison et al. (2010) [60], is an improved N3 bias correction method and it is recognised for its robust performance. As a first step in our data preprocessing pipeline, we make use of N4ITK for bias field correction.

Secondly, MRI does not have a consistent intensity scale, e.g. pixel values of identical tissue, taken in two separate scans in different environments, might not result in the same intensity value. Hence, in prepossessing MRI data the images need to be normalised and brought to the same scale. We normalise every slice to the interval $0 \leq X \leq 1$ with equation 4.1

$$Y = \frac{Y - min(Y)}{max(Y) - min(Y)} \tag{4.1}$$

After bias field correction and normalisation, as described above, we resize the image to the same resolution. Further, we crop each slices to 128x128 to remove uninteresting background regions and make every slice fit our proposed network architecture. Lastly, we perform histogram matching on BraTS data using random subjects from the Cam-CAN dataset. Histogram matching reduces the differences between both datasets. This is important since we make use of both datasets simultaneously in our method.

4.1.3 Evaluation Metrics

In order to compare the proposed methods to other baselines we chose to evaluate them all with the quantitative metrics area under the ROC curve (AUC) [12] and Dice score [20]. AUC measures the total area under the ROC curve, which plots the true positive rate against the false positive rate. AUC is in no need of a binary segmentation map, since it is invariant of the segmentation thresholds and rather measuring the quality of the models prediction. Dice score, as defined in Equation 3.9, on the other hand needs a binary segmentation map for measuring the quality by calculating the degree of spatial overlap between the prediction and ground truth. In medical imaging both AUC and especially Dice score is extensively used as evaluation metrics, and hence they are chosen to evaluate this work.

4.1.4 Implementation Details

The proposed methods was implemented using pytorch and full project code repository is available at `https://github.com/jkkronk/semisupervised_lesion_seg/`. All models were trained using a batch size of 32 on Nvidia Titan X and Xp GPUs.

4.2 Results for Baselines

4.2.1 Unsupervised Lesion Detection via Image Restoration with a Normative Prior

Our method is an extension of the method proposed by You et al. naturally we compare our results to this method. Keeping in mind that You et al. only leverage unlabelled data and the likelihood term is modelled with TV norm, then our extension should at least improve its performance.

Differently to the proposed methods from You et al. we trained the VAE, for learning the normative prior, with data augmentation. BraTS dataset and Cam-CAN dataset have some differences, such as sharpness, contrast and intensities, and data augmentation while training the VAE can help to better reconstruct images from BraTS dataset. We used an identical VAE to approximate the normative prior in both the proposed methods and the unsupervised restoration. The VAE was trained, with the unlabelled "healthy" dataset Cam-CAN, using Adam optimiser with learning rate 10^{-3}. The training continued for 300 epochs until the validation set started to diverge, which took about 40h. In the restoration process, each slice was restored with 250 steps, with a step size of 3×10^{-3}. The λ hyperparameter, governing the influence of the TV norm, was empirically set to 2.

We visualise the results from our implementation of unsupervised lesion detection in Figure 4.1. It is quite impressive that an unsupervised method like this can detect lesions quite well without having any prior knowledge about them. However, in some cases the methods fails and produce a grossly wrong segmentation. When tested on our dataset, out implementation of the unsupervised method performed AUC $= 0.80$, Dice$(FPR - 0.01) = 0.34$ and Dice score of around 0.35. This performance is comparable to the performance You et al. presented.

4.2.2 U-Net

The implemented U-Net has the same architecture as the original U-Net but. However, different data augmentation was used to increase generalisation of the network and to better suit the problem of brain tumour segmentation. The network was trained using Adam optimiser with learning rate of 10^{-4} and early stop when the validation set performance started to decrease. Training the U-Net took about 2-8h depending on how much training data was available. The detailed architecture of the network can be found in Appendix Section A.1. Visualisation of the resulting segmentation map can be seen in Figure 4.2. We notice that the resulting segmentation looks quite well already when using only 10 subjects. Full test results for U-Net can be found in Appendix Table A.3.

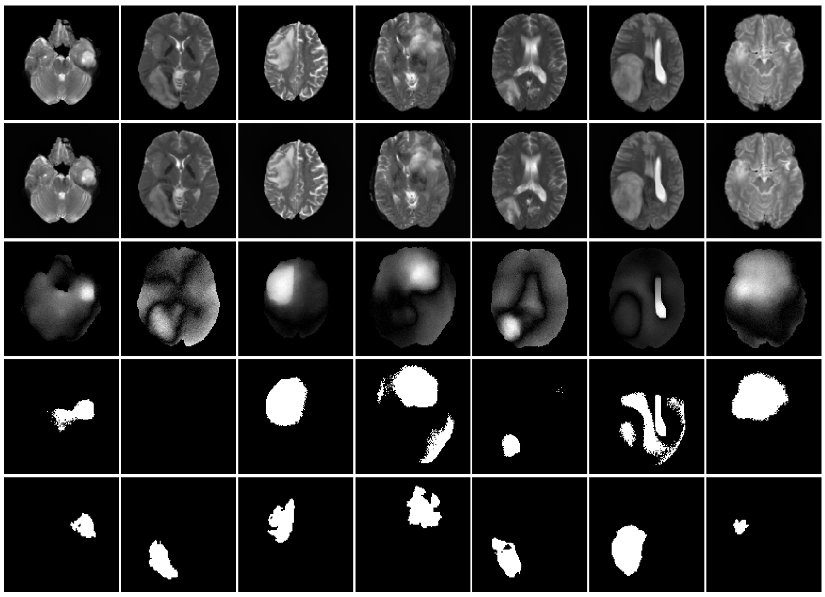


Figure 4.1: Visualisation over restoration method using unsupervised lesion detection [64]. Row 1: Original image; row 2: Restored pseudo "healthy" image; row 3: Resulting segmentation map; row 4 Binary segmentation map using false positive rate 0.01; row 5: Ground truth segmentation

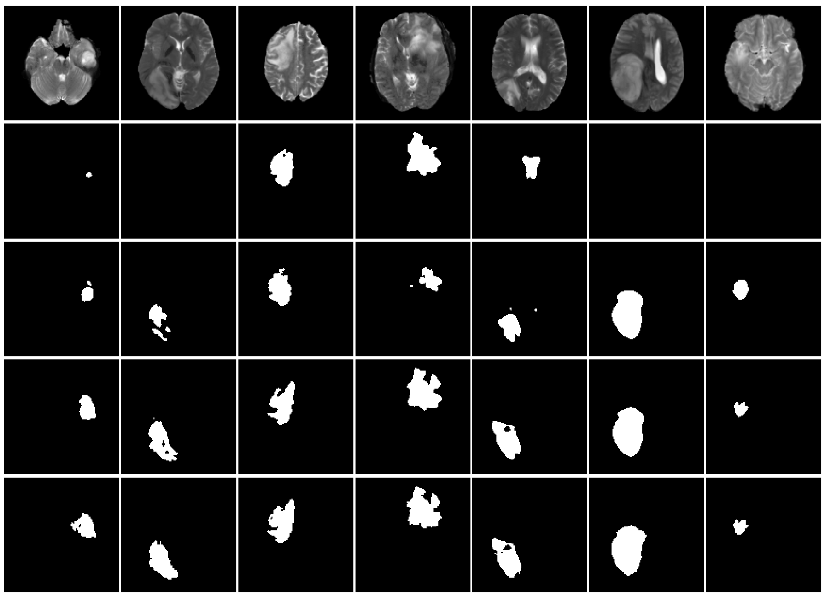


Figure 4.2: Segmentation result using U-Net. Row 1: Original image; row 2: Segmentation using 1 subject; row 3: Segmentation using 10 subject subjects; row 4: Segmentation using 10 subjects; row 5: Segmentation using 100 subjects; row 6: Ground truth segmentation

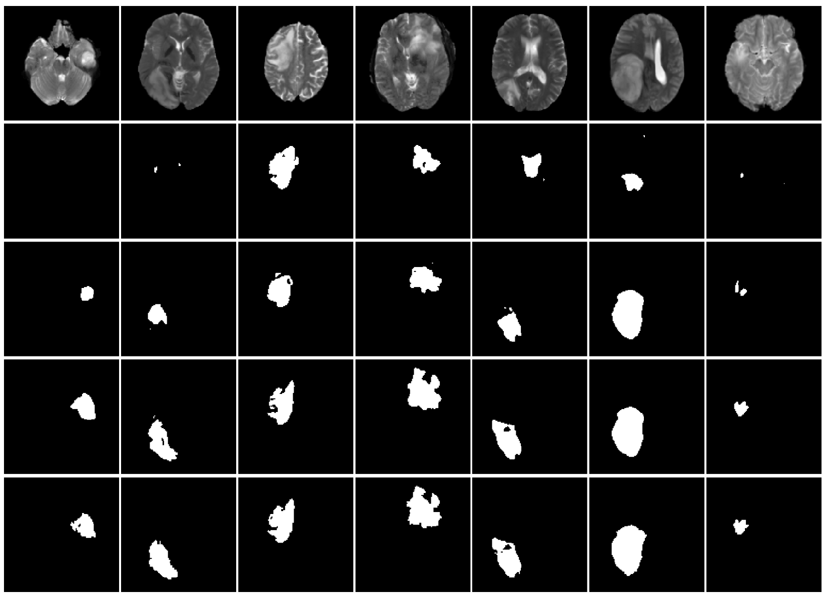


Figure 4.3: Segmentation results using a pretrained U-Net with simCLR. Row 1: Original image; row 2: Segmentation using 1 subject; row 3: Segmentation using 10 subject subjects; row 4: Segmentation using 10 subjects; row 5: Segmentation using 100 subject; row 6: Ground truth segmentation

4.2.3 SimCLR / U-Net - Contrastive Learning

To compare our proposed methods with a baseline that also leverage the unlabelled data, provided in our setting, we implemented simCLR to pretrain the encoder part of the U-Net. The idea is that after the pretraining, the encoder parameters are better initialised and fine tuning the encoder and decoder part, as an original U-Net, should result in at least slightly better performance. The network architecture was identical to the baseline U-Net. We used the original git repository [`https://github.com/sthalles/SimCLR`] for implementation of contrastive loss. Furthermore, we used dot product as similarity function and the contrastive learning was trained using Adam optimiser with learning rate of 10^{-4} and weight decay of 10^{-6}. Early stop was used and the pre-trained model was saved when the validation loss was minimised. The U-Net training parameters were identical to when training without simCLR. Visualisation of the resulting segmentation map can be seen in Figure 4.3. The segmentation results can easily be compared to the results from a vanilla U-Net and its hard to say if there are any improvements from only looking at the segmentations. Full test results for pre-trained U-Net using simCLR can be found in Appendix Table A.4.

4.3 Results for proposed methods

The training for the two methods proposed in this thesis have two separate steps. First, a VAE was trained to approximate a normative prior. The VAE was trained identically as in the baseline method for unsupervised lesion detection, described in Section 4.2.1. Secondly, the segmentation as was trained, in a supervised setting, to approximate the likelihood term. After the training, the restoration and segmentation of test images was performed.

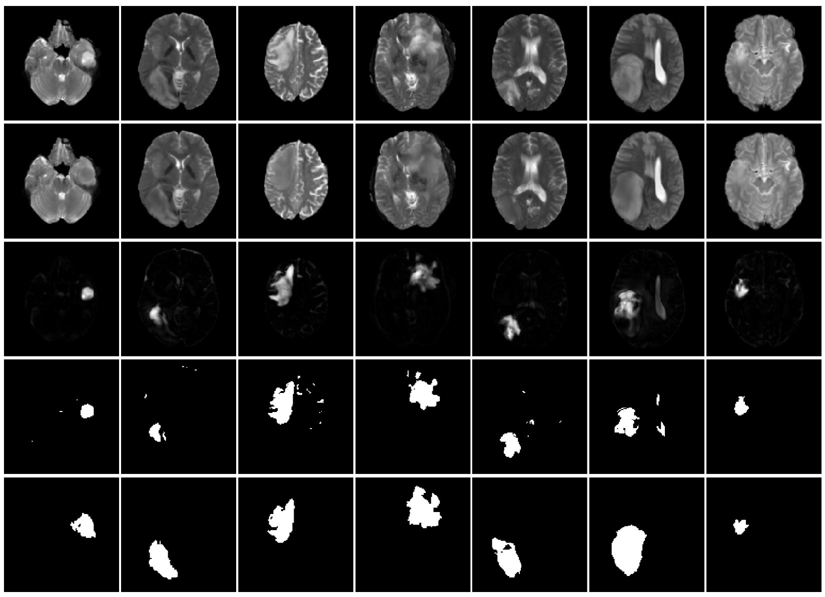


Figure 4.4: Visualisation over restoration method using implicit learning and 1 subject. Row 1: Original image; row 2: Restored pseudo "healthy" image; row 3: Segmentation map; row 4: Binary Segmentation with $fpr = 0.01$; row 5: Ground truth segmentation

4.3.1 Results for Implicit Learning

The segmentation network was trained using Adam optimiser with learning rate 10^{-3} for 100-200 epochs and early stop when validation loss started to increase. Training took about 6-18h, depending on how many subjects were trained. The hyperparameters for restoration were empirically set to $n = 10$ steps with a step size of $\alpha = 3 \times 10^{-1}$. We noticed that increasing the number of restoration steps did not increase the performance, it only increased training time. The step size was set to yield best results and reconstruct tumours as much as possible. Full test results for the implicit approach can be found in Appendix Table A.1.

Figures 4.4, 4.5 and 4.6 visualise the result from the implicit approach using 1, 10 and 100 subjects for training. We notice that the method is pretty good at restoring a lesion image to a pseudo "healthy" counterpart, at least for an untrained eye. When adding more subjects to the training set both the restoration and segmentation seem to get more accurate. Additionally, we can notice that, when trained with only a few subjects, the implicit method is having problem when there are bight ventricles present in the image. Segmentation performance, especially using only few subjects typically results in more false positives.

4.3.2 Results for Explicit Learning

The setup to train the segmentation network with the explicit learning approach is similar to the implicit approach, but with a different loss function. The parameters for the restoration were changed, to step size of 1×10^{-1}, to allow better training using the explicit loss function. The K factor parameter for the explicit loss activation function was set empirically with the help of an extra validation subject. We chose a $K = 100$, as it maximises the performance for the extra validation subject. Similar to training with the implicit loss, the network parameters were optimised using Adam optimiser with a learning rate of 10^{-3} and early stop to

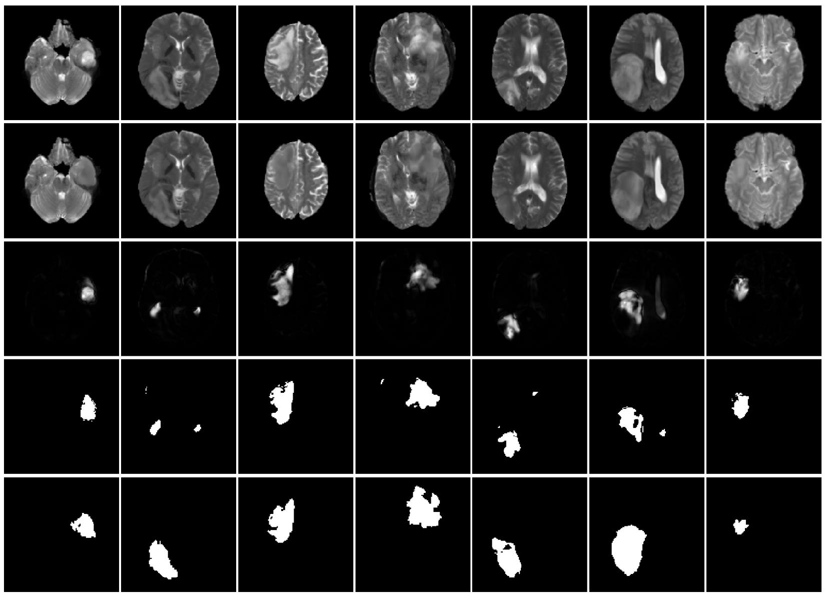


Figure 4.5: Visualisation over restoration method using implicit learning and 10 subjects. Row 1: Original image; row 2: Restored pseudo "healthy" image; row 3: Segmentation map; row 4: Binary Segmentation with $fpr = 0.01$; row 5: Ground truth segmentation

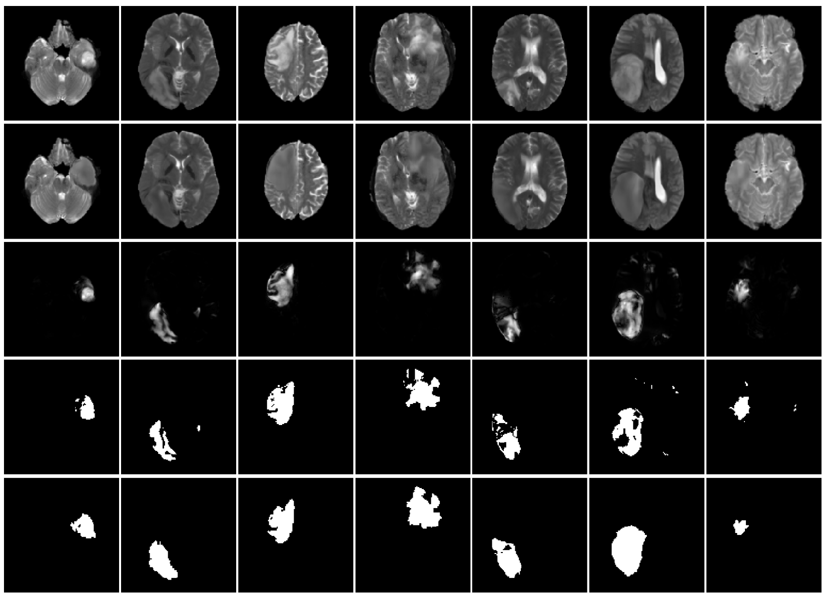


Figure 4.6: Visualisation over restoration method using implicit learning and 100 subjects. Row 1: Original image; row 2: Restored pseudo "healthy" image; row 3: Segmentation map; row 4: Binary Segmentation with $fpr = 0.01$; row 5: Ground truth segmentation

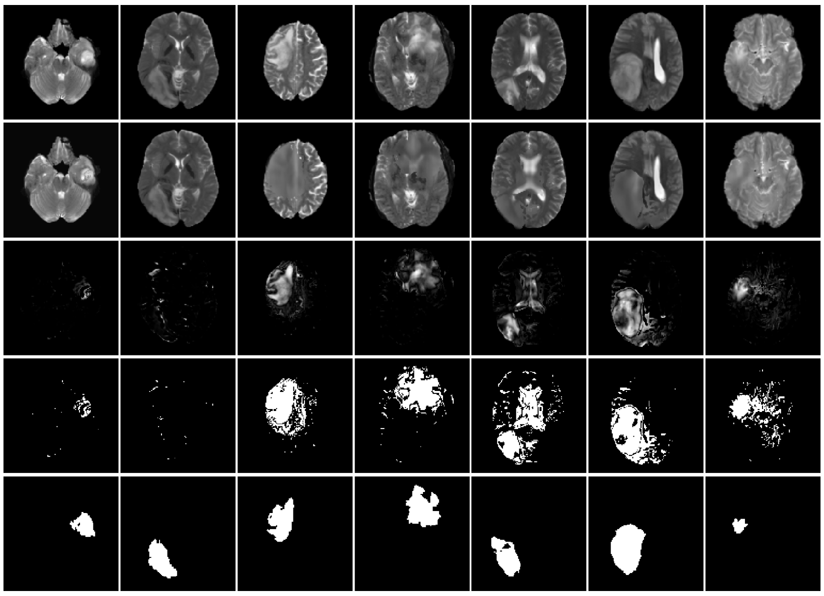


Figure 4.7: Visualisation over restoration method using explicit learning and 1 subjects and $K = 1000$. Row 1: Original image; row 2: Restored pseudo "healthy" image; row 3: Segmentation map; row 4: Binary Segmentation with $fpr = 0.01$; row 5: Ground truth segmentation

minimise overfit.

Figures 4.7, 4.8 and 4.9 visualise the result from the explicit approach using 1, 10 and 100 subjects for training. Firstly, we can see that the method is not working well for neither synthesising a pseudo "healthy" image nor performing a good segmentation. When trained on only one subject there are considerable amount of false positives and the method also to some extent misses tumours completely. Full test results for the explicit learning can be found in Appendix Table A.2.

4.4 Quantitative Comparison

Table 4.1 presents a quantitative comparison between the methods with different number of subjects. Models, trained with 30 subjects or less, is presented with mean scores from testing 5 different models with different training subjects. We trained our models on 1/3/10/30/100 subjects to compare performance. Additionally, Figures 4.10 and 4.11 show proposed and baseline methods compared to number of subjects. Firstly, we notice that already with use of one subject the performance of our methods are increased, compared to the performance of the unsupervised lesion detection method proposed by You et al. Moreover, while using only one subject we show that our method have slightly better performance than a simCLR and a vanilla U-Net. However, already with 3 subject or more the U-Net performs as good or better than our methods. We also notice that the simCLR approach only increase the Dice score slightly when used for pre-training the U-Net. We clearly see a distinct difference in results between implicit and explicit learning approach. The implicit approach outperforms the explicit approach by good margin and we can see that the explicit approach can not compete with baseline methods.

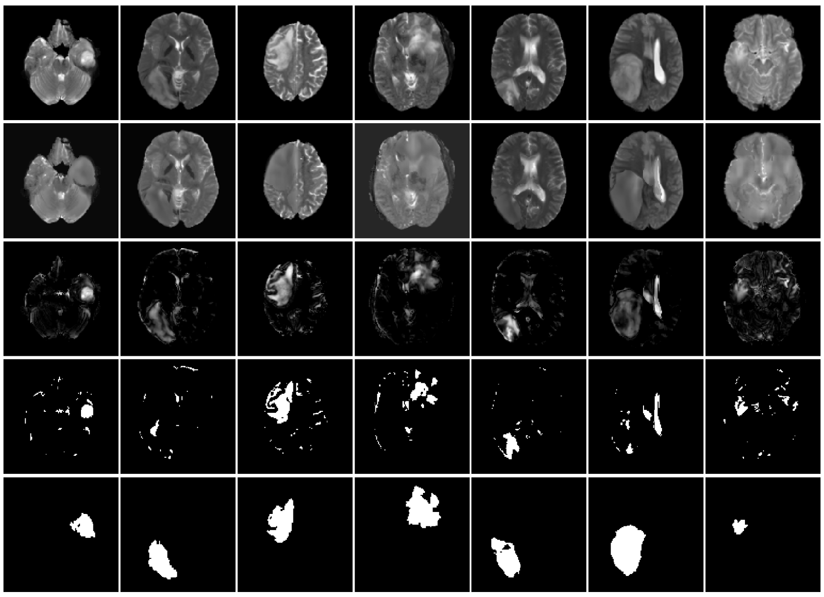


Figure 4.8: Visualisation over restoration method using explicit learning and 10 subjects. Row 1: Original image; row 2: Restored pseudo "healthy" image; row 3: Segmentation map; row 4: Binary Segmentation with $fpr = 0.01$; row 5: Ground truth segmentation

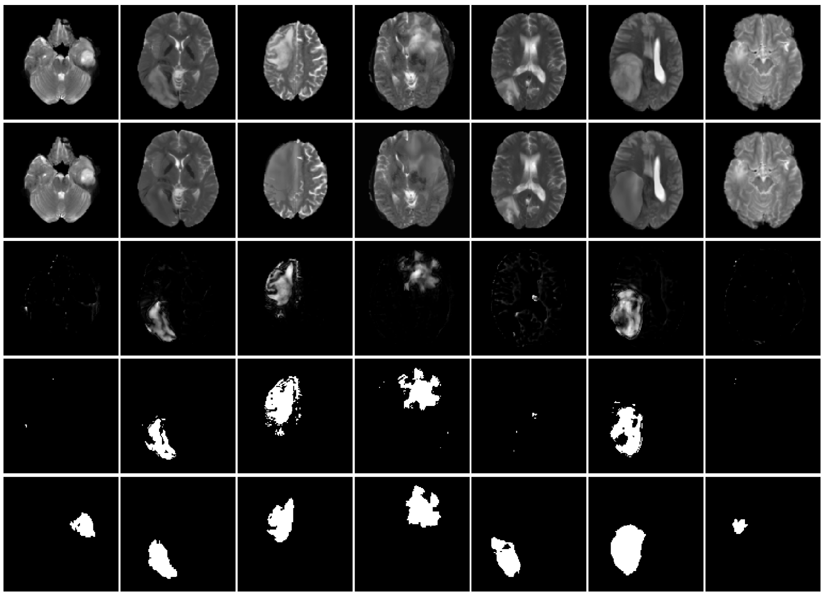


Figure 4.9: Visualisation over restoration method using explicit learning and 100 subjects. Row 1: Original image; row 2: Restored pseudo "healthy" image; row 3: Segmentation map; row 4: Binary Segmentation with $fpr = 0.01$; row 5: Ground truth segmentation

Methods	**AUC**	**Dice (naive)**	**DICE (fpr 0.01)**	**DICE (fpr 0.05)**
You et al.	0.80	–	0.34	0.35

(a) Result comparison using 0 subjects.

Methods	**AUC**	**Dice (naive)**	**Dice (fpr 0.05)**	**Dice (fpr 0.01)**
Implicit Learning	$\mathbf{0.86}(\pm\mathbf{0.02})$	$0.51(\pm0.05)$	$0.51(\pm0.07)$	$\mathbf{0.53}(\pm\mathbf{0.05})$
Explicit Learning *	$0.80(\pm0.04)$	$0.38(\pm0.03)$	$0.41(\pm0.04)$	$0.41(\pm0.03)$
U-Net	$0.85(\pm0.02)$	$0.49(\pm0.06)$	–	–
SimCLR / U-Net	$0.84(\pm0.02)$	$0.50(\pm0.13)$	–	–

(b) Result comparison using 1 subject. * Uses one extra labelled subject for tuning the K-factor.

Methods	**AUC**	**Dice (naive)**	**Dice (fpr 0.05)**	**Dice (fpr 0.01)**
Implicit Learning	$\mathbf{0.90}(\pm\mathbf{0.02})$	$0.61(\pm0.03)$	$0.61(\pm0.05)$	$\mathbf{0.63}(\pm\mathbf{0.02})$
Explicit Learning	$0.83(\pm0.05)$	$0.48(\pm0.09)$	$0.45(\pm0.07)$	$0.47(\pm0.07)$
U-Net	$0.90(\pm0.02)$	$0.58(\pm0.07)$	–	–
SimCLR / U-Net	$0.87(\pm0.05)$	$0.59(\pm0.07)$	–	–

(c) Result comparison using 3 subjects.

Methods	**AUC**	**Dice (naive)**	**Dice (fpr 0.05)**	**Dice (fpr 0.01)**
Implicit Learning	$0.91(\pm0.03)$	$0.64(\pm0.03)$	$0.54(\pm0.12)$	$0.64(\pm0.05)$
Explicit Learning	$0.85(\pm0.04)$	$0.55(\pm0.04)$	$0.45(\pm0.07)$	$0.48(\pm0.06)$
U-Net	$0.94(\pm0.01)$	$0.70(\pm0.03)$	–	–
SimCLR / U-Net	$\mathbf{0.94}(\pm\mathbf{0.01})$	$\mathbf{0.70}(\pm\mathbf{0.04})$	–	–

(d) Result comparison using 10 subjects.

Methods	**AUC**	**Dice (naive)**	**Dice (fpr 0.05)**	**Dice (fpr 0.01)**
Implicit Learning	$0.95(\pm0.01)$	$0.67(\pm0.04)$	$0.65(\pm0.07)$	$0.67(\pm0.03)$
Explicit Learning	$0.88(\pm0.02)$	$0.56(\pm0.04)$	$0.46(\pm0.11)$	$0.49(\pm0.10)$
U-Net	$0.97(\pm0.01)$	$0.78(\pm0.02)$	–	–
SimCLR / U-Net	$\mathbf{0.98}(\pm\mathbf{0.01})$	$\mathbf{0.79}(\pm\mathbf{0.01})$	–	–

(e) Result comparison using 30 subjects.

Methods	**AUC**	**Dice (naive)**	**Dice (fpr 0.05)**	**Dice (fpr 0.01)**
Implicit Learning	0.96	0.73	0.67	0.75
Explicit Learning	0.90	0.61	0.41	0.61
U-Net	**0.98**	**0.82**	–	–
SimCLR / U-Net	**0.98**	**0.82**	–	–

(f) Result comparison using 100 subjects.

Table 4.1: Summarised result comparison between proposed and baselines methods. Dice (naive) corresponds to choosing threshold via the training set for implicit learning and setting the threshold to 0.5 for U-Net and SimCLR.

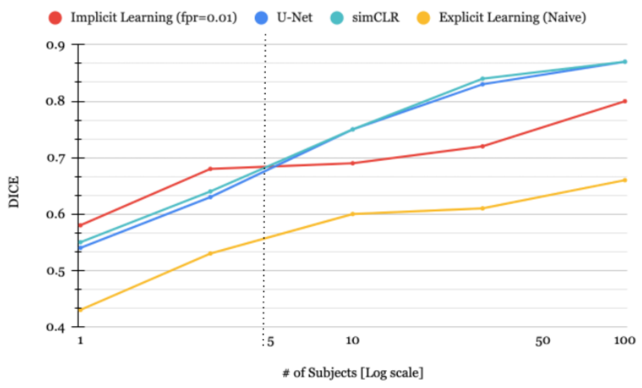


Figure 4.10: Plot showing Dice score depending on subjects for the proposed methods and baselines.

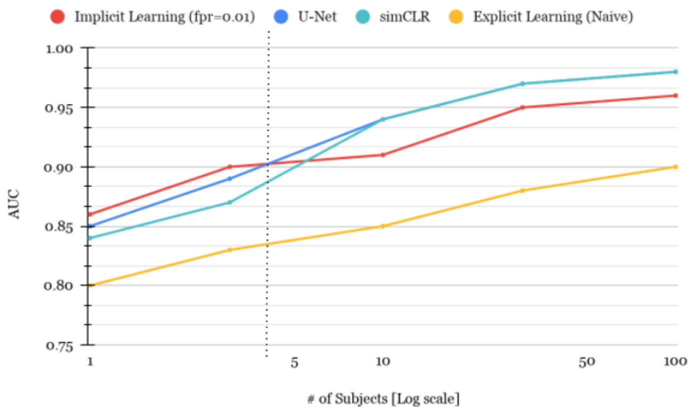


Figure 4.11: Plot showing AUC score depending on subjects for the proposed methods and baselines.

Chapter 5

Discussion and Conclusions

5.1 Discussion

The results show that the proposed semi-supervised method can compete with both unsupervised and supervised baselines methods, especially when only limited labelled training data is available. We were able to show that the implicit approach for learning the likelihood achieves our goal and results in better performance than the unsupervised baseline method of You et al. Additionally, this method can compete with supervised U-Net and semi-supervised U-Net baselines while using less than 10 subjects for training. There is further work to fully leverage the performance of the proposed methods, however they already show potential and initial results are promising. Fortunately, we suggest that there are some improvements that could already be implemented. These improvements are further explained in Section 5.2.

Of the two proposed approaches for learning the likelihood, the implicit approach shows more potential than the explicit approach. Before discarding the explicit approach, other aspects should further be explored, such as fully understanding the K hyper parameter and how to set it to achieve higher performance. We do not yet fully know how sensitive the explicit approach is to changes in K and from our experience, a correct K factor can achieve as high or higher performance as the implicit approach. Nevertheless, as the results indicate, a badly chosen K factor can have devastating results on performance and choosing a K factor empirically with a validation set is shown not to be a good solution. Although, as the implicit approach both performs better and is more simple, any further investigation should first focus on the implicit approach before the explicit approach.

Further implications of this work is coming from a byproduct of our method. As we restore lesion tissue to pseudo "healthy" tissue we create a pseudo "healthy" image from the original brain. This can be useful in clinical practice when a clinician wants to analyse healthy anatomical structures of a brain scan with lesions. Today, most methods for automatic segmentation of healthy tissue typically fail when there is a lesion present in the scan. If our method can produce an image where the tumour is replaced with "healthy" tissue these automatic segmentation models might gain in performance. This subtask of synthesising a pseudo "healthy" brain scan can be compared to the method of Xia et al. and it would be interesting to compare the results between our method and theirs.

One interesting finding can be seen in the restored pseudo "healthy" images shown by Figure 5.1. Firstly, we see in the left images that the front left horn of the lateral ventricle is being restored quite faithfully and fits in the surrounding tissue. It even matches the front right ventricle which is quite impressive. Secondly, in the right images, we see an impressive restoration of the upper ventricles. This shows the potential and strength of the VAE for restoring the image to a pseudo "healthy" counterpart. However, these restorations

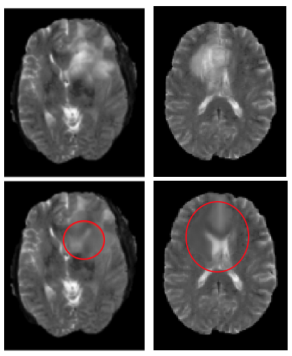


Figure 5.1: Restoration using implicit method with 1 subject. Two lesion images (top) and corresponding restored pseudo ”healthy” images (bottom).

also show that the restored tissue is typically more blurry than normal healthy tissue, which is a common problem when working with VAE. The blurry restoration can be clearly seen when restoring grey matter, where typically our model smears out details too much. Nevertheless, better approximations of the prior, using for example GMVAE or GANs, might solve this issue.

Another interesting finding is visualised in Figure 5.2. Here we see that even if the restoration looks faithful, the corresponding segmentation is not accurate enough. The results suggest that lesion regions have the same intensity after and before restoration. This might indicate that assuming a lesion as additive noise, $D = |Y - X_n|$, might not be completely accurate. Additionally, assuming the lesion as additive noise neglects deformation from the lesion in healthy regions. When lesions, for example, squeeze or deform healthy tissue to something that looks abnormal our method might fail. There might be better assumptions for the lesion, such as seeing the problem as an ill posed problem as $Y = T(X_n) + \delta Y$. However, this would make restoration a lot harder and more complicated to solve, it might even be impossible to find a solution. Even if our approximation of the lesion is not solid, we show that it might be good enough to achieve a good segmentation.

Furthermore, we notice that the segmentation results does not increase as much as baselines U-Net when more subjects are added. This probably originates in the U-Net generalises better with more subjects, however our proposed method is restricted by the prior approximation which does not get better when more subjects is introduced.

5.2 Future Work

To get a better understanding and be able to improve our proposed method, additional future work is needed. In summary, key points listed below can be further explored/improved. The key points are explained in this section.

- Tuning of hyper parameters for restoration.
- Explore other network architecture and/or semi-supervised training of the segmentation network.
- Better approximation of prior, using for example GMVAE.

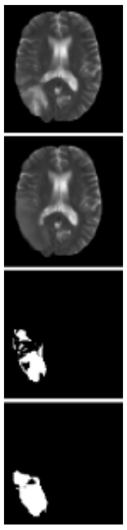


Figure 5.2: Restoration using implicit method with 100 subjects. From top to bottom: Input image; Restored image; Segmentation ($fpr = 0.01$); Ground truth segmentation.

- Extension using 3d volumetric data and/or more MRI sequences.
- Exploration of how to choose K hyperparameter for the explicit learning approach.
- Method tested on different segmentation problem.

Firstly, the understanding of how hyperparameters, such as step size and number of steps in restoration, affects the method's performance needs to be improved. The hyperparameters were empirically set with a validation set, but full evaluation was left out for future work. There might be hyperparameters that show a more robust and better performance. For example, we did not separate the hyperparameters for restoration and training in this work. Empirically, we showed that 10 steps usually were best for training performance. However, during testing the restoration process was restoring more bright lesions, that needed more iterations to converge. At 10 steps all lesions in the test set were not fully restored. We suggest improving this by for example using 10 steps for training and as many steps required until convergence is reached for testing. It would be interesting to see how much such an approach would increase performance.

Secondly, the network architecture for the segmentation network was chosen because of its previously shown performance for segmentation tasks. However, in our case we might gain performance by choosing another architecture that suits our method better. Choosing the best network architecture is a research topic by itself, however, it is quite easy in our implementation to evaluate different architectures directly. For example, one segmentation network architecture that might perform well is [28]. Additionally, it would also be possible to leverage the unlabelled data to train the segmentation network in a semi-supervised setting, with for example simCLR or mean teacher. This would theoretically yield a better segmentation in restoration and hence also improve segmentation performance. Furthermore, a better model for approximating the normative prior would potentially have a great effect on our method. As a first step, it would be interesting to see how implementing a GMVAE instead of VAE would improve performance.

Thirdly, an extension of the proposed method, including more MRI sequences and/or volumetric 3d data would be interesting to explore. Even if there first are some problems to solve before implementing these extensions, it potentially will have a positive effect on the performance. Since the 2d slice segmentation problem we use is a simplified version of the natural 3d volume segmentation problem from MRI, there might be valuable dependencies in the 3d data that we miss. Additionally, the use of more MRI sequences, such as T1-weighted, might help the segmentation network to improve. Today, most state of the art methods for brain lesion segmentation leverages 3d inputs and all sequences. This suggests that our method would also perform better by doing so.

A further investigation on a better approach to find the best performing K factor for the explicit loss function is needed. As the approach is quite promising in theory but failing on the K factor, finding a way to choose the K factor or come around it by choosing another activation function would potentially lead to higher performance. Additionally, it would be interesting to see how the explicit approach performs if trained with a connected graph, as mentioned in Section 3.3, where backpropagation is only performed at the end of the restoration. This might yield a better performance and a more stable training.

Lastly, it would be interesting to understand how the method performs on different datasets and see how well the method works in other environments both for different segmentation problems and other computer vision problems. One example the proposed method potentially would work on is image inpainting or anomaly detection. Additionally, if the method is being used as a restoration method to find pseudo ”healthy” images it would be interesting to find a metric that evaluates how faithful the restoration is and also test it against benchmarks such as Xia et al. This would help improvements for a more accurate restoration and hence also improvements for segmentation results.

5.3 Conclusions

We conclude that our proposed method is promising, but there are still improvements left to be done before reaching state of the art. Even though the problem was found harder than initially thought, we show that turning the unsupervised method of You et al. to a semi-supervised extension can improve performance and even compete with supervised and semi-supervised baselines. Given the list of potential improvements and future work this thesis can be seen as a foundation for a future method that might show even better results.

Appendix A

The First Appendix

A.1 U-Net Architecture

```
UNET(
  (encoder1): Sequential(
    (enc1conv1): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (enc1norm1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (enc1relu1): ReLU(inplace=True)
    (enc1conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (enc1norm2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (enc1relu2): ReLU(inplace=True)
  )
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (encoder2): Sequential(
    (enc2conv1): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (enc2norm1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (enc2relu1): ReLU(inplace=True)
    (enc2conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (enc2norm2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (enc2relu2): ReLU(inplace=True)
  )
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (encoder3): Sequential(
    (enc3conv1): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (enc3norm1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (enc3relu1): ReLU(inplace=True)
    (enc3conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (enc3norm2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (enc3relu2): ReLU(inplace=True)
  )
  (pool3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (encoder4): Sequential(
    (enc4conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (enc4norm1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (enc4relu1): ReLU(inplace=True)
    (enc4conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (enc4norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (enc4relu2): ReLU(inplace=True)
  )
  (pool4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (bottleneck): Sequential(
    (bottleneckconv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bottlenecknorm1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (bottleneckrelu1): ReLU(inplace=True)
    (bottleneckconv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bottlenecknorm2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (bottleneckrelu2): ReLU(inplace=True)
  )
  (upconv4): ConvTranspose2d(256, 128, kernel_size=(2, 2), stride=(2, 2))
  (decoder4): Sequential(
    (dec4conv1): Conv2d(256, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (dec4norm1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (dec4relu1): ReLU(inplace=True)
    (dec4conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (dec4norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (dec4relu2): ReLU(inplace=True)
  )
  (upconv3): ConvTranspose2d(128, 64, kernel_size=(2, 2), stride=(2, 2))
  (decoder3): Sequential(
    (dec3conv1): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (dec3norm1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (dec3relu1): ReLU(inplace=True)
    (dec3conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (dec3norm2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (dec3relu2): ReLU(inplace=True)
  )
  (upconv2): ConvTranspose2d(64, 32, kernel_size=(2, 2), stride=(2, 2))
  (decoder2): Sequential(
    (dec2conv1): Conv2d(64, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (dec2norm1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (dec2relu1): ReLU(inplace=True)
    (dec2conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (dec2norm2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (dec2relu2): ReLU(inplace=True)
  )
  (upconv1): ConvTranspose2d(32, 16, kernel_size=(2, 2), stride=(2, 2))
  (decoder1): Sequential(
    (dec1conv1): Conv2d(32, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (dec1norm1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (dec1relu1): ReLU(inplace=True)
    (dec1conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (dec1norm2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (dec1relu2): ReLU(inplace=True)
  )
  (conv): Conv2d(16, 1, kernel_size=(1, 1), stride=(1, 1))
)
```

A.2 Shallow U-Net Architecture

```
shallow_UNet(
  (encoder1): Sequential(
    (enc1conv1): Conv2d(2, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (enc1norm1): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (enc1relu1): ReLU(inplace=True)
    (enc1conv2): Conv2d(4, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (enc1norm2): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (enc1relu2): ReLU(inplace=True)
  )
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (encoder2): Sequential(
    (enc2conv1): Conv2d(4, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (enc2norm1): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (enc2relu1): ReLU(inplace=True)
    (enc2conv2): Conv2d(8, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (enc2norm2): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (enc2relu2): ReLU(inplace=True)
  )
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (encoder3): Sequential(
    (enc2conv1): Conv2d(8, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (enc2norm1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (enc2relu1): ReLU(inplace=True)
    (enc2conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (enc2norm2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (enc2relu2): ReLU(inplace=True)
  )
  (pool3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (bottleneck): Sequential(
    (bottleneckconv1): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bottlenecknorm1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (bottleneckrelu1): ReLU(inplace=True)
    (bottleneckconv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bottlenecknorm2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (bottleneckrelu2): ReLU(inplace=True)
  )
  (upconv3): Sequential(
    (0): Upsample(scale_factor=4.0, mode=bilinear)
    (1): Conv2d(32, 16, kernel_size=(2, 2), stride=(2, 2))
  )
  (decoder3): Sequential(
    (dec2conv1): Conv2d(32, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (dec2norm1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (dec2relu1): ReLU(inplace=True)
    (dec2conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (dec2norm2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (dec2relu2): ReLU(inplace=True)
  )
  (upconv2): Sequential(
    (0): Upsample(scale_factor=4.0, mode=bilinear)
    (1): Conv2d(16, 8, kernel_size=(2, 2), stride=(2, 2))
  )
  (decoder2): Sequential(
    (dec2conv1): Conv2d(16, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (dec2norm1): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (dec2relu1): ReLU(inplace=True)
    (dec2conv2): Conv2d(8, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (dec2norm2): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (dec2relu2): ReLU(inplace=True)
  )
  (upconv1): Sequential(
    (0): Upsample(scale_factor=4.0, mode=bilinear)
    (1): Conv2d(8, 4, kernel_size=(2, 2), stride=(2, 2))
  )
  (decoder1): Sequential(
    (dec1conv1): Conv2d(8, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (dec1norm1): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (dec1relu1): ReLU(inplace=True)
    (dec1conv2): Conv2d(4, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (dec1norm2): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (dec1relu2): ReLU(inplace=True)
  )
  (conv): Conv2d(4, 1, kernel_size=(1, 1), stride=(1, 1))
```

A.3 VAE Architecture

```
ConvVAE(
  (encoder): Sequential(
    (0): Sequential(
      (0): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (1): ResBlock_Down(
      (conv1): Conv2d(16, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
      (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential(
        (0): Conv2d(16, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): LeakyReLU(negative_slope=0.2)
      )
    )
    (2): ResBlock_Down(
      (conv1): Conv2d(16, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
      (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential(
        (0): Conv2d(16, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): LeakyReLU(negative_slope=0.2)
      )
    )
    (3): ResBlock_Down(
      (conv1): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
      (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential(
        (0): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): LeakyReLU(negative_slope=0.2)
      )
    )
    (4): ResBlock_Down(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential(
        (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): LeakyReLU(negative_slope=0.2)
      )
    )
    (5): ResBlock_Down(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential(
        (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): LeakyReLU(negative_slope=0.2)
      )
    )
  )
  (res_encoder): Sequential(
    (0): Sequential(
      (0): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

```
        (2): LeakyReLU(negative_slope=0.2)
      )
      (1): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(2, 2), dilation=(2, 2))
      (2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (3): LeakyReLU(negative_slope=0.2)
      (4): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(2, 2), dilation=(2, 2))
      (5): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (6): LeakyReLU(negative_slope=0.2)
      (7): Conv2d(32, 16, kernel_size=(3, 3), stride=(1, 1), padding=(2, 2), dilation=(2, 2))
      (8): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (9): LeakyReLU(negative_slope=0.2)
      (10): Conv2d(16, 1, kernel_size=(3, 3), stride=(1, 1), padding=(2, 2), dilation=(2, 2))
    )
    (fc1): ResBlock_Down(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential(
        (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (fc2): ResBlock_Down(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential(
        (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (decoder): Sequential(
      (0): ResBlock_Up(
        (conv1): Sequential(
          (0): Upsample(scale_factor=2.0, mode=bilinear)
          (1): Conv2d(512, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        )
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (conv2): Sequential(
          (0): Upsample(scale_factor=2.0, mode=bilinear)
          (1): Conv2d(512, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        )
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (shortcut): Sequential(
          (0): Upsample(scale_factor=4.0, mode=bilinear)
          (1): Conv2d(512, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
          (2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (1): ResBlock_Up(
        (conv1): Sequential(
          (0): Upsample(scale_factor=2.0, mode=bilinear)
          (1): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        )
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (conv2): Sequential(
          (0): Upsample(scale_factor=2.0, mode=bilinear)
          (1): Conv2d(256, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        )
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (shortcut): Sequential(
          (0): Upsample(scale_factor=4.0, mode=bilinear)
          (1): Conv2d(256, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
          (2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (2): ResBlock_Up(
        (conv1): Sequential(
          (0): Upsample(scale_factor=2.0, mode=bilinear)
          (1): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        )
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (conv2): Sequential(
          (0): Upsample(scale_factor=2.0, mode=bilinear)
          (1): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        )
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (shortcut): Sequential(
          (0): Upsample(scale_factor=4.0, mode=bilinear)
          (1): Conv2d(128, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
          (2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (3): ResBlock_Up(
        (conv1): Sequential(
          (0): Upsample(scale_factor=2.0, mode=bilinear)
          (1): Conv2d(64, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        )
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (conv2): Sequential(
          (0): Upsample(scale_factor=2.0, mode=bilinear)
          (1): Conv2d(64, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        )
        (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (shortcut): Sequential(
          (0): Upsample(scale_factor=4.0, mode=bilinear)
          (1): Conv2d(64, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
          (2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (4): ResBlock_Up(
        (conv1): Sequential(
          (0): Upsample(scale_factor=2.0, mode=bilinear)
          (1): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        )
        (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (conv2): Sequential(
          (0): Upsample(scale_factor=2.0, mode=bilinear)
          (1): Conv2d(32, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        )
        (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (shortcut): Sequential(
          (0): Upsample(scale_factor=4.0, mode=bilinear)
          (1): Conv2d(32, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
          (2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (5): ResBlock_Up(
        (conv1): Sequential(
          (0): Upsample(scale_factor=2.0, mode=bilinear)
          (1): Conv2d(16, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        )
        (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (conv2): Sequential(
          (0): Upsample(scale_factor=2.0, mode=bilinear)
          (1): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        )
        (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (shortcut): Sequential(
          (0): Upsample(scale_factor=4.0, mode=bilinear)
          (1): Conv2d(16, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
          (2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (6): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (7): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (8): LeakyReLU(negative_slope=0.2)
      (9): Conv2d(16, 1, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    )
  )
```

A.4 Full Results

Table A.1: Full test results using implicit learning method.

Number of Training Subjects	AUC	Dice (naive)	Dice (fpr 0.05)	Dice (fpr 0.01)
1	0.87	0.48	0.49	0.49
1	0.88	0.59	0.60	0.60
1	0.84	0.49	0.55	0.56
1	0.86	0.52	0.50	0.53
1	0.85	0.46	0.42	0.47
3	0.89	0.59	0.64	0.63
3	0.92	0.65	0.64	0.66
3	0.91	0.59	0.51	0.62
3	0.88	0.58	0.62	0.62
3	0.91	0.62	0.62	0.62
10	0.87	0.60	0.43	0.59
10	0.94	0.63	0.63	0.63
10	0.94	0.68	0.68	0.71
10	0.91	0.64	0.64	0.65
10	0.88	0.63	0.63	0.60
30	0.94	0.63	0.64	0.64
30	0.96	0.71	0.71	0.71
30	0.94	0.63	0.64	0.64
30	0.95	0.67	0.55	0.68
30	0.95	0.71	0.72	0.70
100	0.96	0.73	0.67	0.75

Table A.2: Full test results using explicit learning method.

Number of Training Subjects	AUC	Dice (naive)	Dice (fpr 0.05)	Dice (fpr 0.01)
1	0.83	0.34	0.38	0.41
1	0.78	0.39	0.40	0.39
1	0.77	0.37	0.46	0.46
1	0.78	0.42	0.42	0.42
1	0.86	0.40	0.37	0.38
3	0.78	0.31	0.36	0.37
3	0.86	0.53	0.44	0.49
3	0.90	0.56	0.56	0.55
3	0.81	0.52	0.47	0.50
3	0.82	0.47	0.44	0.46
10	0.78	0.52	0.34	0.38
10	0.84	0.58	0.46	0.51
10	0.87	0.59	0.45	0.50
10	0.86	0.54	0.51	0.51
10	0.89	0.51	0.49	0.50
30	0.88	0.57	0.31	0.40
30	0.88	0.63	0.62	0.63
30	0.84	0.51	0.40	0.40
30	0.90	0.55	0.47	0.48
30	0.90	0.56	0.48	0.53
100	0.90	0.61	0.43	0.61

Table A.3: Full test results using U-Net baselines.

Number of Training Subjects	**AUC**	**Dice**
1	0.84	0.44
1	0.83	0.52
1	0.86	0.56
1	0.85	0.41
1	0.89	0.50
3	0.88	0.57
3	0.92	0.66
3	0.89	0.53
3	0.87	0.50
3	0.92	0.66
10	0.94	0.69
10	0.93	0.67
10	0.95	0.74
10	0.95	0.71
10	0.93	0.67
30	0.96	0.76
30	0.97	0.79
30	0.97	0.77
30	0.97	0.77
30	0.98	0.80
100	0.98	0.82

Table A.4: Full test results using pre-trained U-Net with simCLR baselines.

Number of Training Subjects	**AUC**	**Dice**
1	0.85	0.48
1	0.85	0.56
1	0.84	0.60
1	0.80	0.32
1	0.84	0.54
3	0.86	0.51
3	0.93	0.70
3	0.87	0.59
3	0.90	0.55
3	0.80	0.59
10	0.94	0.70
10	0.95	0.69
10	0.94	0.74
10	0.96	0.73
10	0.92	0.64
30	0.97	0.77
30	0.97	0.80
30	0.98	0.80
30	0.97	0.78
30	0.98	0.79
100	0.98	0.82

Bibliography

[1] Jonas Adler and Ozan Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, Nov 2017.

[2] Mohammed Sabbih Hamoud Al-Tamimi and Ghazali Sulong. Tumor brain detection through mr images: A review of literature. *Journal of Theoretical & Applied Information Technology*, 62(2), 2014.

[3] Kenneth Aldape, Kevin M Brindle, Louis Chesler, Rajesh Chopra, Amar Gajjar, Mark R Gilbert, Nicholas Gottardo, David H Gutmann, Darren Hargrave, Eric C Holland, et al. Challenges to curing primary brain tumours. *Nature Reviews Clinical Oncology*, 16(8):509–520, 2019.

[4] Safaa E Amin and MA Megeed. Brain tumor diagnosis systems based on artificial neural networks and segmentation using mri. In *2012 8th International Conference on Informatics and Systems (INFOS)*, pages MM–119. IEEE, 2012.

[5] Elsa D Angelini, Olivier Clatz, Emmanuel Mandonnet, Ender Konukoglu, Laurent Capelle, and Hugues Duffau. Glioma dynamics and computational models: a review of segmentation, registration, and in silico growth algorithms and their clinical applications. *Current Medical Imaging*, 3(4):262–276, 2007.

[6] Wenjia Bai, Ozan Oktay, Matthew Sinclair, Hideaki Suzuki, Martin Rajchl, Giacomo Tarroni, Ben Glocker, Andrew King, Paul M. Matthews, and Daniel Rueckert. Semi-supervised learning for network-based cardiac mr image segmentation. In Maxime Descoteaux, Lena Maier-Hein, Alfred Franz, Pierre Jannin, D. Louis Collins, and Simon Duchesne, editors, *Medical Image Computing and Computer-Assisted Intervention MICCAI 2017*, pages 253–260, Cham, 2017. Springer International Publishing.

[7] Spyridon Bakas, Mauricio Reyes, András Jakab, Stefan Bauer, Markus Rempfler, Alessandro Crimi, Russell Shinohara, Christoph Berger, Sung Ha, Martin Rozycki, Marcel Prastawa, Esther Alberts, Jana Lipkova, John Freymann, Justin Kirby, Michel Bilello, Hassan Fathallah-Shaykh, Roland Wiest, and Jan Kirschke. Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the brats challenge. 11 2018.

[8] Stefan Bauer, Roland Wiest, Lutz-P Nolte, and Mauricio Reyes. A survey of mri-based medical image analysis for brain tumor studies. *Physics in Medicine & Biology*, 58(13):R97, 2013.

[9] Christoph Baur, Shadi Albarqouni, and Nassir Navab. Semi-supervised deep learning for fully convolutional networks. In Maxime Descoteaux, Lena Maier-Hein, Alfred Franz, Pierre Jannin, D. Louis Collins, and Simon Duchesne, editors, *Medical Image Computing and Computer Assisted Intervention MICCAI 2017*, pages 311–319, Cham, 2017. Springer International Publishing.

[10] Christoph Baur, Benedikt Wiestler, Shadi Albarqouni, and Nassir Navab. Deep autoencoding models for unsupervised anomaly segmentation in brain mr images. In *International MICCAI Brainlesion Workshop*, pages 161–169. Springer, 2018.

[11] Christopher Bowles, Chen Qin, Ricardo Guerrero, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna Wardlaw, and Daniel Rueckert. Brain lesion segmentation through image synthesis and outlier detection. *NeuroImage: Clinical*, 16:643–658, 2017.

[12] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[13] Sreenivasa R Chandana, Sujana Movva, Madan Arora, and Trevor Singh. Primary brain tumors in adults. *American family physician*, 77(10):1423, 2008.

[14] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[15] Xiaoran Chen and Ender Konukoglu. Unsupervised detection of lesions in brain mri using constrained adversarial auto-encoders. *arXiv preprint arXiv:1806.04972*, 2018.

[16] Xiaoran Chen, Suhang You, Kerem Can Tezcan, and Ender Konukoglu. Unsupervised lesion detection via image restoration with a normative prior. *Medical Image Analysis*, page 101713, 2020.

[17] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pages 424–432. Springer, 2016.

[18] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3642–3649. IEEE, 2012.

[19] Lisa M. DeAngelis. Brain tumors. *New England Journal of Medicine*, 344(2):114–123, 2001. PMID: 11150363.

[20] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[21] Nat Dilokthanakul, Pedro Mediano, Marta Garnelo, Matthew Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. 11 2016.

[22] Hao Dong, Guang Yang, Fangde Liu, Yuanhan Mo, and Yike Guo. Automatic brain tumor detection and segmentation using u-net based fully convolutional networks. In *annual conference on medical image understanding and analysis*, pages 506–517. Springer, 2017.

[23] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

[24] Guray Erus, Harsha Battapady, Theodore D Satterthwaite, Hakon Hakonarson, Raquel E Gur, Christos Davatzikos, and Ruben C Gur. Imaging patterns of brain development and their relationship to cognition. *Cerebral Cortex*, 25(6):1676–1684, 2015.

[25] Peter Gibbs, David L Buckley, Stephen J Blackband, and Anthony Horsman. Tumour volume determination from mr images by morphological segmentation. *Physics in Medicine & Biology*, 41(11):2437, 1996.

[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[27] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[28] Nabil Ibtehaz and M Sohel Rahman. Multiresunet: Rethinking the u-net architecture for multimodal biomedical image segmentation. *Neural Networks*, 121:74–87, 2020.

[29] Fabian Isensee, Philipp Kickingereder, Wolfgang Wick, Martin Bendszus, and Klaus H Maier-Hein. No new-net. In *International MICCAI Brainlesion Workshop*, pages 234–244. Springer, 2018.

[30] A. G. Ivakhnenko. Polynomial theory of complex systems. *IEEE Trans. Syst. Man Cybern.*, 1:364–378, 1971.

[31] Vasileios G Kanas, Evangelia I Zacharaki, Evangelos Dermatas, Anastasios Bezerianos, Kyriakos Sgarbas, and Christos Davatzikos. Combining outlier detection with random walker for automatic brain tumor segmentation. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 26–35. Springer, 2012.

[32] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[33] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[34] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.

[35] Ender Konukoglu and Ben Glocker. Subcmap: Subject and condition specific effect maps. *CoRR*, abs/1701.02610, 2017.

[36] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[37] Stanley Osher Leonid I. Rudin and Emad Fatemi. Nonlinear total variation based noise removal algorithms. 1992.

[38] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[39] Raphael Meier, Stefan Bauer, Johannes Slotboom, Roland Wiest, and Mauricio Reyes. Patient-specific semi-supervised learning for postoperative brain tumor segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 714–721. Springer, 2014.

[40] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, L. Lanczi, E. Gerstner, M. Weber, T. Arbel, B. B. Avants, N. Ayache, P. Buendia, D. L. Collins, N. Cordier, J. J. Corso, A. Criminisi, T. Das, H. Delingette, Ç. Demiralp, C. R. Durst, M. Dojat, S. Doyle, J. Festa, F. Forbes, E. Geremia, B. Glocker, P. Golland, X. Guo, A. Hamamci, K. M. Iftekharuddin, R. Jena, N. M. John, E. Konukoglu, D. Lashkari, J. A. Mariz, R. Meier, S. Pereira, D. Precup, S. J. Price, T. R. Raviv, S. M. S. Reza, M. Ryan, D. Sarikaya, L. Schwartz, H. Shin, J. Shotton, C. A. Silva, N. Sousa, N. K. Subbanna, G. Szekely, T. J. Taylor, O. M. Thomas, N. J. Tustison, G. Unal, F. Vasseur, M. Wintermark, D. H. Ye, L. Zhao, B. Zhao, D. Zikic, M. Prastawa, M. Reyes, and K. Van Leemput. The multimodal brain tumor image segmentation benchmark (brats). *IEEE Transactions on Medical Imaging*, 34(10):1993–2024, 2015.

[41] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571. IEEE, 2016.

[42] Andriy Myronenko. 3d mri brain tumor segmentation using autoencoder regularization. In *International MICCAI Brainlesion Workshop*, pages 311–320. Springer, 2018.

[43] Jakub Nalepa, Michal Marcinkiewicz, and Michal Kawulok. Data augmentation for brain-tumor segmentation: A review. *Frontiers in Computational Neuroscience*, 13, 2019.

[44] Hiroko Ohgaki and Paul Kleihues. Population-based studies on incidence, survival rates, and genetic alterations in astrocytic and oligodendroglial gliomas. *Journal of neuropathology and experimental neurology*, 64:479–89, 06 2005.

[45] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in neural information processing systems*, pages 3235–3246, 2018.

[46] Nick Pawlowski, Matthew C. H. Lee, Martin Rajchl, Steven McDonagh, Enzo Ferrante, Konstantinos Kamnitsas, Sam Cooke, Susan K. Stevenson, Aneesh Khetani, Tom Newman, Frederick A. Zeiler, Richard John Digby, Jonathan Peter Coles, Daniel Rueckert, David K. Menon, Virginia F. J. Newcombe, and Ben Glocker. Unsupervised lesion detection in brain ct using bayesian convolutional autoencoders. 2018.

[47] Sérgio Pereira, Adriano Pinto, Victor Alves, and Carlos A Silva. Brain tumor segmentation using convolutional neural networks in mri images. *IEEE transactions on medical imaging*, 35(5):1240–1251, 2016.

[48] Christian S. Perone and Julien Cohen-Adad. Deep semi-supervised segmentation with weight-averaged consistency targets. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 12–19, Cham, 2018. Springer International Publishing.

[49] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[50] Herbert E. Robbins. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 2007.

[51] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[52] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[53] Lothar R Schad, Stefan Blüml, and Ivan Zuna. Ix. mr tissue characterization of intracranial tumors by means of texture analysis. *Magnetic resonance imaging*, 11(6):889–896, 1993.

[54] Meredith Shafto, Lorraine Tyler, Marie Dixon, Jason Taylor, James Rowe, Rhodri Cusack, Andrew Calder, William Marslen-Wilson, John Duncan, Tim Dalgleish, Richard Henson, Carol Brayne, and Fiona Matthews. The cambridge centre for ageing and neuroscience (cam-can) study protocol: A cross-sectional, lifespan, multidisciplinary examination of healthy cognitive ageing. *BMC neurology*, 14:204, 10 2014.

[55] Daljit Singh and Kamaljeet Kaur. Classification of abnormalities in brain mri images using glcm, pca and svm. *International Journal of Engineering and Advanced Technology (IJEAT)*, 1(6):243–248, 2012.

[56] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.

[57] Kerem C Tezcan, Christian F Baumgartner, Roger Luechinger, Klaas P Pruessmann, and Ender Konukoglu. Mr image reconstruction using deep density priors. *IEEE transactions on medical imaging*, 38(7):1633–1642, 2018.

[58] Majda Thurnher. The 2007 who classification of tumors of the central nervous system – what has changed? *American Journal of Neuroradiology*, 01 2012.

[59] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[60] Nicholas J Tustison, Brian B Avants, Philip A Cook, Yuanjie Zheng, Alexander Egan, Paul A Yushkevich, and James C Gee. N4itk: improved n3 bias correction. *IEEE transactions on medical imaging*, 29(6):1310–1320, 2010.

[61] M Vaidyanathan, LP Clarke, RP Velthuizen, S Phuphanich, AM Bensaid, LO Hall, JC Bezdek, H Greenberg, A Trotti, and M Silbiger. Comparison of supervised mri segmentation methods for tumor volume determination during therapy. *Magnetic resonance imaging*, 13(5):719–728, 1995.

[62] B. Wang, K. W. Liu, K. M. Prastawa, A. Irima, P. M. Vespa, J. D. van Horn, P. T. Fletcher, and G. Gerig. 4d active cut: An interactive tool for pathological anatomy modeling. In *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, pages 529–532, 2014.

[63] Tian Xia, Agisilaos Chartsias, and Sotirios A Tsaftaris. Pseudo-healthy synthesis with pathology disentanglement and adversarial learning. *arXiv preprint arXiv:2005.01607*, 2020.

[64] Suhang You, Kerem C. Tezcan, Xiaoran Chen, and Ender Konukoglu. Unsupervised lesion detection via image restoration with a normative prior. In M. Jorge Cardoso, Aasa Feragen, Ben Glocker, Ender Konukoglu, Ipek Oguz, Gozde Unal, and Tom Vercauteren, editors, *Proceedings of the 2nd International Conference on Medical Imaging with Deep Learning (MIDL 2019)*, volume 109 of *Proceedings of Machine Learning Research*, pages 540 – 556, Cambridge, MA, 2019. PMLR. 2nd International Conference on Medical Imaging with Deep Learning (MIDL 2019); Conference Location: London, United Kingdom; Conference Date: July 8-10, 2019.

[65] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.