Semi-Supervised Medical Image Segmentation with Equivariance Regularization

Gustav Schölin

Master's thesis
2020:E18

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Semi-Supervised Medical Image Segmentation with Equivariance Regularization

Gustav Schölin
tfy14gsc@student.lu.se

Principal Advisor: Karl Åström (karl.astrom@math.lth.se)

Assistant Advisor: Kenneth Batstone (kenneth_john.batstone@math.lth.se)

Company Advisors: Åsa Bertze (asa@peltarion.com),
Paolo Elena (paolo@peltarion.com)

Examiner: Niels Christian Overgaard (niels_christian.overgaard@math.lth.se)

April 20, 2020

Abstract

The last decades of research in machine learning and deep learning have lead to enormous advancements in the field. One of the areas that stand to gain the most from this is the medical sector. However, the majority of deep learning models today rely on supervised learning and one considerable bottleneck in the sector's ability to adapt the technology is the need of large labeled datasets. Inspired by newly published semi-supervised methods for image classification, this work addresses the problem for the task of semantic segmentation (a task that is recurrent in medical imaging and cancer treatment) by introducing a semi-supervised method, named Equivariance Regularization (EquiReg). Using the EquiReg-method the model is trained to output equivariant predictions with respect to data augmentations using unlabeled data, in conjunction with standard supervised training using labeled data. Experiments with brain tumor MRI-scans from the BraTS 2019 dataset show that the EquiReg method can, when only a small percentage of data is labeled, boost performance by incorporating unlabeled data during training. Furthermore, the experiments show that the EquiReg-method also improves performance in the fully supervised case, by using the labeled data for both supervised and unsupervised training of the same model.

Preface

This thesis project was done during the fall of 2019 at the Centre for Mathematical Sciences at Lund University and together with Peltarion AB. All the work was conducted at Peltarion's offices in Stockholm. I would like to thank my industry supervisors, Åsa Bertze and Paolo Elena at Peltarion, for all guidance and support during the work. I would also like to direct special appreciation to everyone at Peltarion for giving me the opportunity to conduct my master thesis there and providing me with an inspiring working environment, many interesting discussions and great learning experiences during this thesis.

Contents

1 Introduction

Modern medicine today is heavily dependent not only on human expertise but also on advanced technology, such as medical imaging. The imaging technology, such as magnetic resonance imaging (MRI) or computed tomography (CT), helps medical staff to perform a better job and provide patients with better healthcare. However, in order for medical imaging to be applied effectively, it is often also required of a medical specialist to put in a lot of work to analyse and interpret the output.

An example of this is when a patient is in need of radiotherapy or some quantitative analysis is to be done from a imaging scan. As a first step in these cases radiologists often need to segment the scan images, i.e., assigning different parts of every image a category, like for example healthy tissue, organ or tumor. In short, telling what is what in the image. However, manual segmentation is tedious work and it can take a radiologist hours for each scan. Automating this task while keeping accuracy, or at the very least semi-automating it giving the radiologist automatically generated suggestion to review, would mean faster diagnoses for patients, higher efficiency for doctors and hopefully more lives saved.

In recent years, as image analysis with deep learning has taken massive steps forward, medical image segmentation has also become a deep learning problem. Although there has been progress in the area, a deep learning model generally requires many thousands annotated data samples for training of the model. This is often problematic when it comes to medical data due to patient confidentiality and related issues. This in combination with the time consuming manual segmentation results in a small amount of labeled training data available.

The limited availability and high cost of labeled examples make a semi-supervised model an appealing approach for this task, i.e., a model that makes use of both labeled and unlabeled data during training. This master thesis project use ideas from semi-supervised image classification to develop such a semi-supervised segmentation model for medical imaging data. To try to answer questions like how the purposed model compares with a fully supervised model and how the semi-supervised model perform with different amounts of labeled data a series of experiments are conducted with a dataset containing MRI-scans of brain tumors.

The outline of this report is as follows. Section 2 will explain the task of semantic segmentation but also give an introduction to Artificial Neural Networks (ANNs) and how they can be used as a tool for the segmentation task while Section 3 covers semi-supervised learning. Further, Section 4 will present a overview of existing research in the area before this work's purposed technique is detailed in Section 5. Section 6 covers the dataset used for training and evaluation of model and the experiments, including details and results, are presented in Section 7. The report is then wrapped up with discussions about the results in Section 8, some suggestions for future work to improve the model in Section 9 and conclusions about the work as a whole in Section 10.

2 Semantic Segmentation

In the field of computer vision there exists a large variety of learning tasks that are desirable to solve. Some of the most common include image classification, object detection and semantic segmentation. In image classification the objective is to label an image depending on its visual content, i.e., outputting one label per image. In object detection the objective is to detect and mark objects in an image, i.e., outputting a number of label-coordinates pairs per image. This work, however, is focusing on the task of semantic segmentation where the objective is to split up an image in humanly understandable parts via a pixelwise classification of a image, i.e., outputting one label per pixel. The number of classification classes N for each pixel can either be $N = 2$ (binary) or $N > 2$ (multi-class). The segmentation problem occurs in many different areas such as processing camera images in autonomous driving, different kinds of detection in aerial- and satellite images or processing medical images.

The segmentation problem can be seen as approximating a function f^* to map an input

image, $\boldsymbol{x}$, to a segmentation map, $\boldsymbol{y}$, as

$$\boldsymbol{y} = f(\boldsymbol{x}), \tag{1}$$

where f is an approximation of f^*. Commonly, $\boldsymbol{x} \in \mathbb{R}^{h \times w \times c}$ and $\boldsymbol{y} \in \{0\ ,\ \ldots\ ,\ N-1\}^{h \times w}$ where h and w is the pixel height and width of the image and c is the number of channels in the input image.

2.1 Artificial Neural Networks

The recent decade's enormous advancements in the field of Artificial Neural Networks (ANNs) in general, and Convolutional Neural Networks (CNNs) in particular, have meant great strides forward for solving most computer vision learning tasks, semantic segmentation included.

Even though the reader is assumed to be familiar with ANNs, a short introduction is given here. A more thorough review of the topic is provided by [11]. ANNs are a class of parameterized functions $f(\boldsymbol{x};\boldsymbol{\theta})$ constructed by composing linear functions and non-linear functions as

$$f = a_n \circ f_n \circ \ldots \circ a_k \circ f_k \circ \ldots \circ a_1 \circ f_1. \tag{2}$$

Here f_k is a linear function parameterized by its weights θ_k and a_k is a non-linear function, referred to as activation function. Common choices of activation functions include the logistic function, $a(x) = \frac{1}{1+e^{-\alpha x}}$, the hyperbolic tangent function, $a(x) = \frac{e^{2x}-1}{e^{2x}+1}$, or the rectifier function (ReLU), $a(x) = max(0, x)$.

Traditionally the composition of a_k and f_k is referred to as a layer and an ANN is commonly built up by a large number of layers as suggested in (2). However, as research in deep learning has progressed the meaning of the term layer has broadened and today there exists a large variety of layer types that, as will be shown in the following subsections, can be made up of different kinds of function compositions. The simpler layers are usually visualized as a graph with two sets of nodes, where every node in the first set represent an input to the layer and every node in the second set represents an output from the layer. The weights of the layer are then visualized as edges between input and output nodes, and depending how these edges are connected the properties of the layer change. The simplest of layers is the fully connected layer, where, as the name suggests, there exists an edge between every input node and every output node, making all layer outputs linear combinations of the layer inputs.

The process of finding appropriate weights for the ANN is seen as an optimization problem and is referred to as training the ANN. Consequently the set of inputs used for training the ANN is referred to as training data. The goal of the training is to minimize a so called loss function with respect to the weights. Broadly, this is done by calculating the gradients of the loss function with respect to the weights and move in the negative gradient direction by updating the weights. The method is called gradient decent and is the foundation of the most popular optimization algorithms for ANNs, such as ADAM [19], RMSprop [30] or Momentum [28]. The set of training data samples pushed through the ANN before each weight update is generally referred to as a training batch.

It can be shown that given big enough layer dimensions ANNs are universal function approximators [13, 9]. This means that an ANN with large enough layer dimensions can approximate any linear or non-linear function arbitrary well. Given this, ANNs are an appropriate tool for approximating the function in (1).

2.1.1 Convolutional Neural Networks

ANNs containing so-called convolutional layers are known as a Convolutional Neural Networks (CNNs) and are frequently used when the neural network input is image data. Unlike the fully connected layer, the convolutional layer is written as

$$l_{conv} = p \circ a \circ f_{conv}, \tag{3}$$

where f_{conv} is a convolution function parameterized by its weights θ, a is an activation function, commonly ReLU or some version of it, and p is a pooling function. The convolution function can be thought of in terms of the aforementioned graph visualization. Using this, the convolution function only allow output nodes to be linear combinations of input nodes close to each other and the weights are shared between output nodes. This is equal to having a weight matrix, called a kernel, sliding over the input, and outputting the dot product between the kernel and the inputs covered by the kernel. It is common for a convolutional layer to use multiple kernels for f_{conv}. The output from each kernel is referred to as a feature map. Note that if the input to the layer consist of multiple feature maps each kernel is sliding over all feature maps at once with a depth equal to the number of feature maps. Hence, the number of output feature maps does not depend on the number of input feature maps and the number of feature maps both have increased or decreased after a convolutional layer.

The pooling function can similarly be thought of as a sliding window over the inputs. The most common pooling function for CNNs is max pooling and outputs the max of the input values covered by the window. The purpose of the pooling function is generally to reduce the spatial dimensions of the layer input.

The earliest and most researched application area for CNNs is image classification. While reducing spatial dimensions with pooling functions these networks also increase the number of feature maps from each layer. After a certain number of convolutional layers, when the spatial dimensions are small and the number of feature maps large, fully connected layers are added eventually generating a label output. Figure 1 illustrates a typical CNN architecture for image classification.

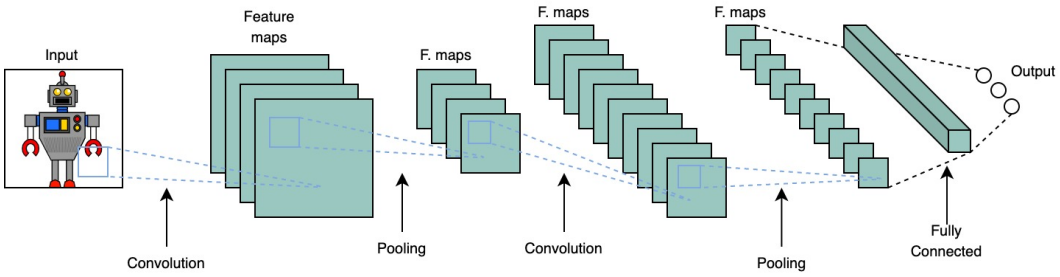


Figure 1: A simplistic figure illustrating a CNN architecture for image classification. Robot image from [27].

One of the strong properties of this kind of CNN architecture is spatial invariance, meaning that the CNN can detect features in the image independent of in which part in the image these features occur.

2.1.2 Fully Convolutional Neural Networks

For the segmentation problem a segmentation map with the same dimensions as the input is desired as output. However, an architecture as in Figure 1 only outputs a single class label. To solve this problem an extended CNN architecture is needed called Fully Convolutional Neural Networks (FCNN) [22, 29].

The way FCNNs differs from classification CNNs is that instead of adding fully connected layers at the end of the network a different kind of layer is added to gradually regain the spatial dimensions of the network input. This type of layer will in this work be referred to as an up-convolution layer. The up-convolution layer can be implemented in two different ways. The first implementation consists of an up-sampling function followed by a normal convolution function as seen in the convolutional layer. The up-sampling function is increasing the spatial dimension via interpolation, for example nearest-neighbor or bilinear.

The second way to implement the up-convolution layer is to use a transposed convolution function. The transposed convolution function can intuitively be thought of as a reversed convolution function even though this is not quite mathematically accurate. Instead of having a kernel sliding over the input, as in a normal convolution, the transposed convolution would equal a kernel sliding over output distributing each input value according to the kernel weights. That is, to each output covered by the kernel window the product between the input and the corresponding weight is added. The spatial dimensions of the output is then controlled by how large strides the kernel takes for each input.

Figure 2 shows an FCNN architecture in a simplified way. The architecture can naturally be split up in two paths beginning with a down-sampling path much like the classification architecture in Figure 1 but without the fully connected layers. The second part is then considered an up-sampling path where the spatial dimensionality is increased step-by-step to finally output a segmentation map of the input.

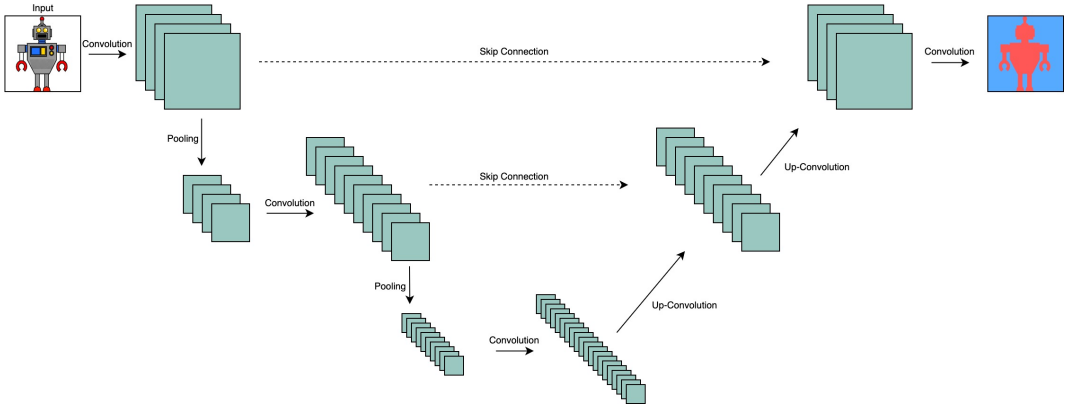


Figure 2: A simplistic figure illustrating a FCNN architecture for image segmentation. Robot image from [27].

However, while down-sampling is good for detecting features in the image it will undeniably also lead to loss of spatial information. While this loss is fine for classification, in segmentation it makes the up-sampling harder consequently affecting the quality of the resulting segmentation maps negatively. To mitigate this, skip connections between the down-convolution path and the up-convolution path are often present in FCNNs. This means that feature maps from the down-convolution path are concatenated with feature maps with the same dimensions from the up-convolution path in order to help preserve spatial information.

2.1.3 Dropout and Batch Normalization

Beyond the aforementioned layer types there are two additional special type of layers used in this work. Those are the Dropout layer [35] and the Batch Normalization layer [16] and will be explained in more detail in this section.

The Dropout layer is known to be a method for regularization of an ANN. If a model is trained too extensively on the training data it can start modeling the residual variance in the training data. This will make the model perform worse on data not belonging to the training data and is referred to as overfitting or overtraining. The goal of regularization is to prevent this and make the model, in this case an ANN, generalize better to the distribution of data from which its training data origins. The way Dropout layers regularize an ANN is by randomly setting a fraction p of the inputs to 0 at each training step. Effectively this means that the connections between the layer before and the layer after the Dropout layer are randomly changed during training. By doing this the idea is to break-up dependencies between connections and make each connection more independent and robust. During evaluation all inputs to the Dropout layer are kept. Hence, in order to keep the sum of the input values to the Dropout layer constant between training and evaluation, each kept input during training is scaled with $\frac{1}{1-p}$.

The purpose of Batch Normalization is to make the training process of an ANN faster and more stable. The problem Batch Normalization sets out to solve is so called covariate shift, meaning that the distribution of the input to each layer in an ANN changes during training. Consequently, the weights in ANN needs to compensate for these distribution changes which makes training slower. To mitigate this problem the idea is to normalize layer inputs, i.e., rescaling to have a mean of 0 and a standard deviation of 1. The normalization is done on each input dimension individually over a batch. However, in order not to constrain the input values, two learnable parameters are added for each input dimension. These parameters scales and shifts the normalized value and are updated along with the other ANN parameters. As an example, the scaling and shifting could undo the normalization if that would be the optimal thing to do in order to minimize the loss function. Beyond reducing covariate shift, there is also some suggestion that adding Batch Normalization layers smooths and, in turn, simplifies the optimization function making the training faster and more stable [32].

3 Semi-Supervised Learning

Machine learning algorithms are traditionally split up in two different categories, supervised and unsupervised, based on the data used during the learning phase. Supervised learning refers to learning some function that maps input data to output data using input-output pairs, i.e., labeled data. This means that supervised machine learning algorithms need a labeled dataset during training. Supervised learning approaches is often used in tasks such as classification and regression.

In unsupervised learning the data used in the learning process is unlabeled. Consequently, unsupervised learning is generally used more in the sense of finding patterns in a dataset rather than doing predictions. Machine learning models that often use unsupervised learning approaches include, among others, clustering models and generative models.

Semi-supervised learning approaches are, as the name suggests, positioned somewhere between supervised and unsupervised learning, meaning that they use both labeled and unlabeled data during the training phase. One common application of semi-supervised learning is in approaching supervised learning tasks, such as classification, combining a small set of labeled data and larger set of unlabeled data. A model using a small amount of labeled data and a larger amount of unlabeled data can be very valuable in tasks where labeled data is expensive to produce in relation to unlabeled data.

The oldest approach to semi-supervised learning is to supervised train the model with the both the labeled and unlabeled data where pseudo-labels for the unlabeled data are provided by the model itself. This method is often referred to as self-training or self-learning. More recent approaches includes for example using generative models for semi-supervised learning [20]. However, the method most related to this work has emerged in recent years and is called Consistency Regularization. The method will be explained in the following section.

3.1 Consistency Regularization

The Consistency Regularization method in semi-supervised learning have been used in a number of works over the past years, primarily for the task of image classification [31, 21, 5, 37]. The main idea is to train the model to be consistent in output predictions given semantic-preserving augmentations of some unlabeled input data. E.g., in an image classification setting, the model is conditioned to produce the same prediction on several examples produced by augmentation (rotation, zooming, etc) of the same unlabeled input image. In the works first purposing Consistency Regularization [31, 21] it is done by minimizing the mean squared difference between the models output from two augmentations of the same unlabeled data sample. This can be written as

$$\|f(\tilde{x}_1) - f(\tilde{x}_2)\|_2^2, \tag{4}$$

where f represents model as a function and $\tilde{x}_1$ and $\tilde{x}_2$ are two augmentations from one unlabeled data sample x.

However, variations of this has come in more recent works, like in [37] where the unsupervised loss is defined as

$$D_{\mathrm{KL}}(f(x)\|f(\tilde{x})), \tag{5}$$

where D_{KL} is the Kullback–Leibler divergence (explained in detail in Section 5.2). Despite the variations in approach the goal is still to make the model predictions invariant to semantic-preserving augmentations applied to input data.

Since Consistency Regularization is a relatively new method there is a lot of research conducted in the area to find the optimal approach.

4 Related Work

General medical image segmentation with deep learning has in recent years mainly been revolving around different types of FCNN architectures. The most well-known work is U-Net [29] which purposed a lot of the architectural designs seen other FCNNs today. E.g., the architectures in [7, 24] builds on U-Net architecture but also leverage 3D-convolutional layers since its common for medical images to have a native 3D-structures. A recent work showing strong results is [26] where a 3D-FCNN is used in combination with a Variational Auto-Encoder (VAE) to better cluster features.

Looking at semi-supervised segmentation one find that it is a relatively unexplored sub-field in machine learning and deep learning. There has, however, been some works published in recent years [34, 15, 25] all using Generative Adversarial Networks (GANs) [12] to solve the problem. GANs are generative models where inspiration has been taken from game theory to have two ANNs competing against each other. Commonly there is one generator ANN which generate fake data from some data distribution and one discriminator ANN which try to distinguish the fake data from real data. All three papers use the GAN setup in slightly different ways and none of them focus on segmentation in the medical image domain, but rather more general benchmark datasets.

However, as mentioned in Section 3 there is a lot of ongoing research in semi-supervised learning focused around image classification, especially using different forms of consistency regularization. Examples of this can be found in [31, 21, 37, 5, 6, 33]. Also in these cases the focus lies on benchmark datasets rather than domain specific data.

5 Semi-Supervised Segmentation

The main idea for this work is to apply semi-supervised learning methods for the task of semantic segmentation. In order to do this a FCNN is trained using a strategy including a variant of consistency regularization for segmentation named Equivariance Regularisation. The following sections will describe the FCNN architecture and the training strategy in detail.

5.1 Fully Convolutional DenseNet (Tiramisu)

The FCNN architecture used in this work is known as a Fully Convolutional DenseNet or Tiramisu [17] and is, as its name suggests, a FCNN extension from the CNN architecture DenseNet [14]. Figure 3a illustrates an overview of the Tiramisu architecture.

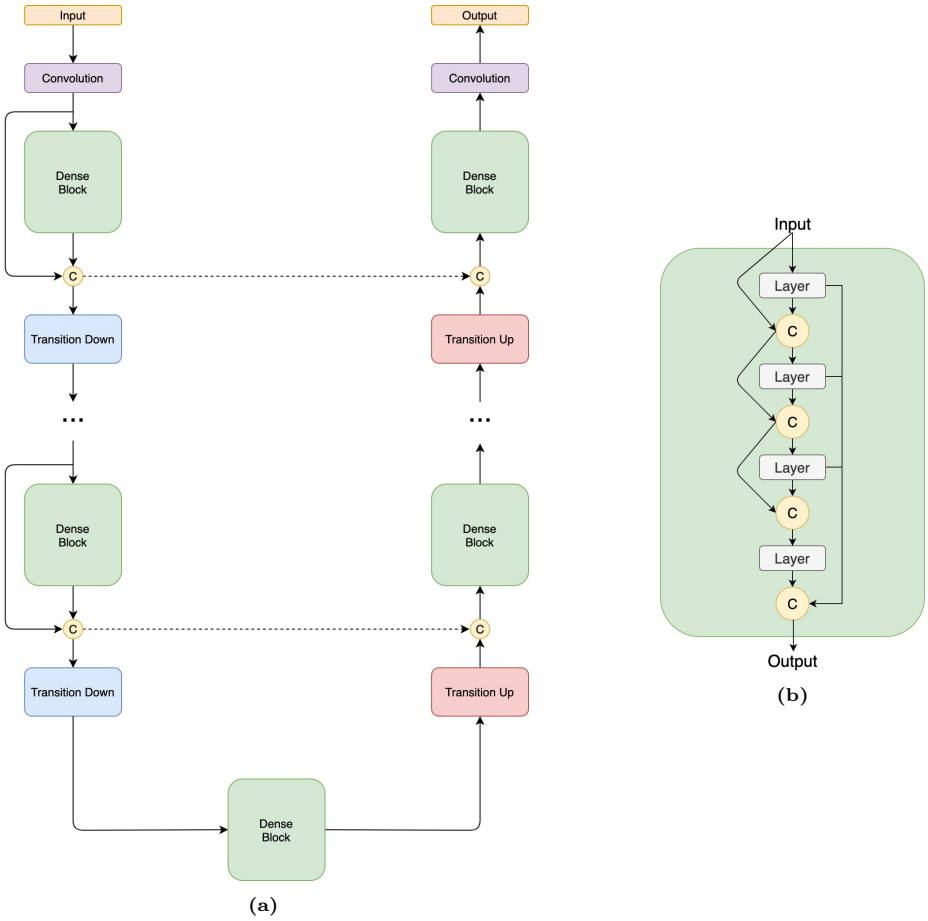


Figure 3: Illustrations of the Tiramisu network architecture. (a) An overview of the architecture. (b) The architecture of the Dense Block.

As can be seen in Figure 3a the Tiramisu architecture is mainly built up from three different components: Dense Blocks, Transition Down layers and Transition Up layers. The unique part of this architecture is the Dense Block, which contains a number of convolutional layers. The input to each layer in the Dense Block is the concatenation of the input feature maps to the block and the output feature maps of all previous layers in the block. The output of the block is the concatenation of all layer outputs in the block. This means that the number of feature maps outputted by the Dense Block is $n \times k$ where n is the number of layers in the Dense Block and k, called growth rate, is the number of output feature maps for each layer. All of this is illustrated in Figure 3b for a Dense Block with 4 layers. Each layer is identical and contains, as specified in Table 1, a batch normalization layer, a ReLU activation function, a convolutional layer with a kernel size of 3×3 and a dropout layer with dropout probability of 0.2.

Table 1: The compositions of the Dense Block, Transition Down and Transition Up layers used in Tiramisu.

Dense Block Layer
Batch Normalization
ReLU
3×3 Convolution
Dropout $p = 0.2$

Transition Down
Batch Normalization
ReLU
1×1 Convolution
Dropout $p = 0.2$
2×2 Max Pooling

Transition Up
3×3 Transposed Convolution (stride $= 2$)

All layers in the Dense Block maintain the spatial dimensions of the input. Dimension reduction and increase happen instead in the Transition Down and Transition Up layers respectively. As Table 1 reveals, the Transition Down layer is similar to the Dense Block

layer with the exception of a 1×1 kernel in the convolutional layer and the addition of a 2×2 max pooling layer in the end. The Transition Up layer solely contains a 3×3 transposed convolution with a stride of 2. The 2×2 kernel of the pooling operation in the Transition Down layer effectively halves the spatial dimensions whereas the stride of the 3×3 transposed convolution in the Transition Up layer doubles them.

As discussed in Section 2.1.2, the Tiramisu architecture also includes skip connections from the down-sampling path to the up-sampling path where the spatial dimension of the feature maps match.

The Tiramisu network used in this work has a total of 15 Dense Blocks, with the number of layers of each block increasing along the down-sampling path and decreasing along the up-sampling path (see Table 2 for details). Note that along the down-sampling path the input to each Dense Block is concatenated with its output in order to grow the number of feature maps. However, this is not done in the up-sampling path, since it would lead to a computationally intractable number of feature maps. This means that each up-sampling Dense Block will have to summarize the information from previous feature maps and will undoubtedly lead to some information loss. The idea, however, is that some of this information can be recovered through the skip connections in the network instead. Table 2 shows the dimensions and number feature maps in different stages of the network for a 224×224 input image.

Table 2: The specific Tiramisu network used in this work, including spatial dimensions and number of feature maps in different stages of the network. Here, *ch* stands for the number channels in the input image and C stands for the number of classes to predict.

Layers	**Spat. Dimensions**	**Nb Feature Maps**
Input	224×224	ch
3×3 Convolution	224×224	48
DB (4 layers) + TD	112×112	112
DB (5 layers) + TD	56×56	192
DB (7 layers) + TD	28×28	304
DB (10 layers) + TD	14×14	464
DB (12 layers) + TD	7×7	656
DB (15 layers)	7×7	240
TU	14×14	896
DB (12 layers) + TU	28×28	704
DB (10 layers) + TU	56×56	496
DB (7 layers) + TU	112×112	352
DB (5 layers) + TU	224×224	224
DB (4 layers)	224×224	64
1×1 Convolution	224×224	C

5.2 Training Strategy

The key part of this work is the semi-supervised training strategy which is visualized in Figure 4. The strategy is in large parts based on that in [37], but modified to work in the semantic segmentation setting instead of the image classification setting. The main idea of the strategy is training the model to not only output correct segmentation maps using labeled images, but also to segment unlabeled images in a equivariant manner with respect to augmentations. Equivariance in this case implies that segmenting and then augmenting would produce the same result as augmenting and then segmenting. Intuitively this is a good property for the model to have since it could make the model more consistent and thereby increase segmentation quality. Furthermore, this is done without the need of additional labeled data but instead utilizing otherwise unusable unlabeled data. Since this new method builds on consistency regularisation but includes equivariance due to the nature of the segmentation problem it is named Equivariance Regularization (EquiReg).

In practice this translates to, in each training step, calculating two different losses. Given a batch containing both labeled and unlabeled training data, one supervised loss based is

calculated using the labeled data and one unsupervised loss is calculated using the unlabeled data. Then, at the end of the training step, the two losses are combined into a final loss.

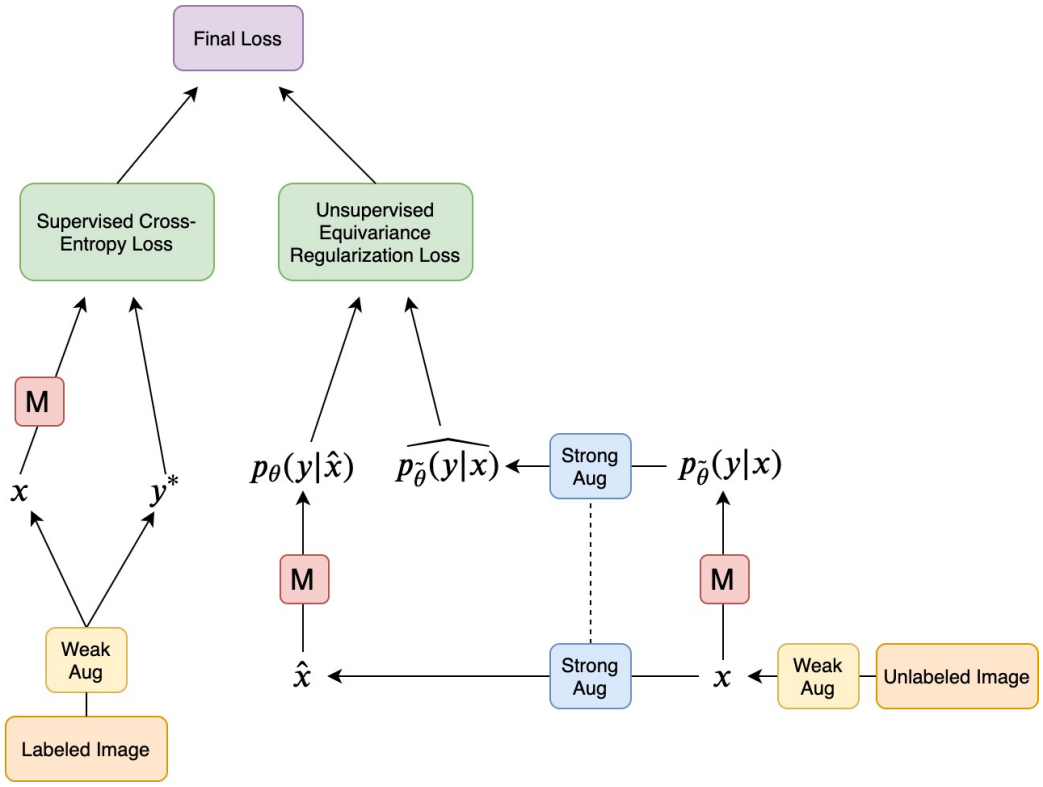


Figure 4: Illustration showing training strategy used in this work, including the unsupervised Equivariance Regularization.

The supervised part of the training strategy starts with a labeled image, i.e., the image x and its corresponding ground truth segmentation map y^*. The model takes x as input and outputs a probability distribution $p_\theta(y_n|x)$ for each pixel n in x. That is, for each pixel n the network outputs C probabilities which sums to 1, where C is the number of possible classes for a pixel. Lets denote $p_\theta(y|x)$ as the set of probability distributions for all pixels in x. Recall that θ represents the weights of the ANN, in this case the Tiramisu network. The ground truth segmentation map y^* is also seen as probability distributions for each pixel but where each pixel's correct class has probability 1 and the rest of the classes has probability 0. The supervised loss is categorical cross-entropy and is calculated from these two sets of probability distributions. Given a pair of discrete probability distributions, u and v, in the same probability space $\mathcal{Z}$ the categorical cross-entropy of v relative to u is written as

$$H(u, v) = -\sum_{z \in \mathcal{Z}} u(z) \log v(z). \tag{6}$$

Given this, the supervised loss can be defined as the mean categorical cross-entropy over all probability distributions of y^* and $p_\theta(y|x)$ paired pixel-wise. This can be written as

$$\mathcal{L}_{sup}(y^*, p_\theta(y|x)) = \frac{1}{N} \sum_{n}^{N} H(y_n^*, y_n) = -\frac{1}{N} \sum_{n}^{N} \sum_{c}^{C} y_{n,c}^* \log(p_\theta(y_{n,c}|x)), \tag{7}$$

where N is the total number of pixels in x.

While the supervised loss is relatively straightforward and commonly used in most areas of machine learning, the unsupervised EquiReg loss is more complex and novel. The main idea of EquiReg is to do a form of consistency regularisation as in [37] but for a segmentation

model using augmentation methods and Kullback–Leibler divergence (KL-divergence). As Figure 4 shows the EquiReg loss calculation starts with a unlabeled image x. The image is copied and one copy is put through the network, giving $p_{\tilde{\theta}}(y|x)$ as a set of probability distributions in the same manner as for the supervised loss. As in [37], $\tilde{\theta}$ represents a fixed version of the current model weights meaning that these weights will not be updated during the gradient decent optimization. The other image copy is augmented, meaning the content of the image is distorted in some or several ways. Examples of image augmentations can be translation, rotation, shearing or scaling. The augmentation schema used in this work is explained in detail in Section 6.2. The augmented image $\hat{x}$ is then also put through the network giving $p_\theta(y|\hat{x})$. Because $p_{\tilde{\theta}}(y|x)$ and $p_\theta(y|\hat{x})$ are sets of probability distributions representing all pixels in x and $\hat{x}$ respectively, pixel-wise corresponding distributions in the sets are not directly comparable. However, if $p_{\tilde{\theta}}(y|x)$ is augmented in the same identical way as $x \to \hat{x}$, giving $\widehat{p_{\tilde{\theta}}(y|x)}$, the pixel-wise corresponding distributions in $\widehat{p_{\tilde{\theta}}(y|x)}$ and $p_\theta(y|\hat{x})$ would represent the same original pixel and therefore be comparable. In Figure 4 the augmentations $x \to \hat{x}$ and $p_{\tilde{\theta}}(y|x) \to \widehat{p_{\tilde{\theta}}(y|x)}$ are connected by a dotted line to represent that they are identical.

The next step is to calculate the unsupervised loss from $\widehat{p_{\tilde{\theta}}(y|x)}$ and $p_\theta(y|\hat{x})$ using the KL-divergence. For a pair of discrete probability distributions, u and v, in the same probability space $\mathcal{Z}$ the KL-divergence of v from u is given by

$$D_{\mathrm{KL}}(u \parallel v) = \sum_{z \in \mathcal{Z}} u(z) \log \left(\frac{u(z)}{v(z)} \right) \tag{8}$$

and can be seen as a measurement on how much v differs from u. Using this, the unsupervised loss can be formulated as how much $p_\theta(y|\hat{x})$ differs from $\widehat{p_{\tilde{\theta}}(y|x)}$, i.e. the KL-divergence of $p_\theta(y|\hat{x})$ from $\widehat{p_{\tilde{\theta}}(y|x)}$. Hence, the loss can be written as

$$\begin{aligned} \mathcal{L}_{ER}(\widehat{p_{\tilde{\theta}}(y|x)}, p_\theta(y|\hat{x})) &= \frac{1}{N} \sum_{n}^{N} D_{\mathrm{KL}} \left(\widehat{p_{\tilde{\theta}}(y_n|x)} \parallel p_\theta(y_n|\hat{x}) \right) \\ &= \frac{1}{N} \sum_{n}^{N} \sum_{c}^{C} \widehat{p_{\tilde{\theta}}(y_{n,c}|x)} \log \left(\frac{\widehat{p_{\tilde{\theta}}(y_{n,c}|x)}}{p_\theta(y_{n,c}|\hat{x})} \right). \end{aligned} \tag{9}$$

Recall that the goal for the EquiReg loss is to make the model equivariant with respect to the augmentations, meaning that segmenting and then augmenting would produce the same result as augmenting and then segmenting. Studying Figure 4 one can see that minimizing the KL-divergence of $p_\theta(y|\hat{x})$ from $\widehat{p_{\tilde{\theta}}(y|x)}$ is doing precisely that. Utilizing unlabeled data for the purpose of equivariance with respect to data augmentation can hopefully make the model more consistent and increase its performance, without using more labeled data.

The last step in the training strategy is to combine the supervised and EquiReg loss into a final loss which then is used in the gradient decent optimization of the models weights. The final loss is formulated as

$$\mathcal{L} = \mathcal{L}_{sup} + \lambda \mathcal{L}_{ER}, \tag{10}$$

where λ is a weighting factor to balance the two parts of the loss. This means the optimization problem is formulated as

$$\min_\theta \mathcal{L}(\theta) = \min_\theta \mathbb{E}_{y^*, x \in \mathrm{L}}[\mathcal{L}_{sup}(y^*, p_\theta(y|x))] + \lambda \mathbb{E}_{x \in \mathrm{U}}[\mathcal{L}_{ER}(\widehat{p_{\tilde{\theta}}(y|x)}, p_\theta(y|\hat{x}))]. \tag{11}$$

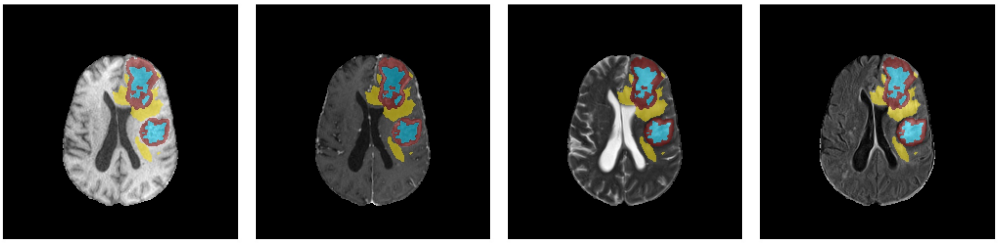
It is also worth noting that both labeled and unlabeled images are weakly augmented right in the beginning of the training strategy, as can be seen in Figure 4. This is a common

method to make sure that the model does not train on the same image more than once if all the training images are looped over multiple times. The weak augmentation is however not as aggressive as the strong one later in the training strategy. Further details are provided in Section 6.2.

6 Data

This work is focused around semi-supervised segmentation for medical images. As mentioned before, there are several different techniques for medical imaging with markedly different characteristics. The image data used in this work was captured using Magnetic Resonance Imaging (MRI), a widely used method to generate images of soft tissue in the body. As the name suggests the technique is built on strong magnetic fields which distinguish it from other X-ray based methods such as computed tomography (CT) and positron-emission tomography (PET). The MRI-dataset was obtained from the Multimodal Brain Tumor Segmentation Challenge 2019 (BraTS 2019) [23, 2, 1, 3, 4] and consists of the 460 pre-operative MRI-scans of human brains suffering from brain tumors. The dataset includes patients with high grade as well as low grade tumors. Of the 460 MRI-scans there are 335 training samples with accompanying ground truth segmentation maps done manually and approved by experienced neuro-radiologists. The remaining 125 scans are test samples and are not released with ground truth segmentation maps, but can still be used to evaluate the performance of a model via uploading the model's predictions to a web portal[1] run by the dataset owners.

A MRI-machine uses radio frequency energy pulses to disrupt the alignment of nuclei in soft tissue caused by a strong magnetic field. In turn, the nuclei emits radio frequency energy when returning to alignment which is detected by the MRI. MRI-scans can display different types of tissue by using different radio frequency pulse sequences. Each scan in this dataset contains 4 different sequences or modalities, T1-weighted (T1), post-contrast T1-weighted (T1C), T2-weighted (T2) and T2-weighted Fluid Attenuated Inversion Recovery (T2-FLAIR). Each sample-scan is a grayscale 3D-image of size $155 \times 240 \times 240$ which means that one sample has the dimensions $4 \times 155 \times 240 \times 240$. Figure 5 shows a 240×240 2D-slice from each modality of a single scan together with the ground truth segmentation map.



(a) T1 **(b)** T1C **(c)** T2 **(d)** T2-FLAIR

Figure 5: 2D-slices of the 4 different modalities together with the ground truth segmentation map from a MRI-scan in the BraTS 2019 dataset.

The segmentation map of each sample in the dataset consists of 4 classes, healthy tissue/background, necrotic and non-enhancing tumor core, peritumoral edema and enhancing tumor. In the BraTS Challenge this 4 class segmentation problem is during evaluation split up in three binary segmentation problems, which is shown in Table 3. The three problems are segmenting the whole tumor, the tumor core and the enhancing tumor where the whole tumor is considered the union of classes 1, 2 and 3, the tumor core is the union of class 1 and 3 and the enhancing tumor is solely class 3.

[1] https://ipp.cbica.upenn.edu/

Table 3: The segmentation classes in the BraTS 2019 dataset as well as the class distribution and the class division for the three binary evaluation problems.

Class	Class Name	Class %	Whole	Core	Enhancing
0	Healthy/Background	98.59%	0	0	0
1	Necrotic and Non-Enhancing	0.32%	1	1	0
2	Peritumoral Edema	0.81%	1	0	0
3	Enhancing	0.28%	1	1	1

The evaluation metric used for these segmentation maps is the Dice score, one of the standard metrics when it comes to segmentation of medical images. The Dice score can be formulated in terms of the set of positives in the prediction X and the set of positives in the ground truth Y but also using true positives (TP), false positives (FP) and false negatives (FN), giving

$$Dice = \frac{2|X \cap Y|}{|X| + |Y|} = \frac{2TP}{2TP + FP + FN}. \tag{12}$$

For a perfect match between prediction and ground truth the Dice score would be 1 and with no intersection between predicted positives and ground truth positives the Dice score would be 0. However, it is worth noting here that the Dice score is undefined when $|X| = |Y| = 0$ or $TP = FP = FN = 0$ which in this setting would correspond to a model that is correctly predicting no tumor at all. To alleviate this potential problem and rewarding the model for correctly predicting no tumor the Dice score is here modified to be

$$Dice = \begin{cases} 1, & \text{if } TP = FP = FN = 0, \\ \frac{2TP}{2TP+FP+FN}, & \text{otherwise.} \end{cases} \tag{13}$$

Looking at Table 3 one can see that the class distribution of the dataset is heavily skewed with large majority of voxels being healthy or background. This is very good reason to use Dice score over for example accuracy since the Dice score is not affected by the number of true negatives. However, even though this is quite natural in the setting of brain tumor segmentation, the tumor is usually small compared to the whole brain, it can potentially pose a problem for the model giving it too few tumor classed voxels to learn to differentiate the classes. One possible solution to this problem would be to introduce a class weighted loss function for the model effectively telling the model to value some classes more than others during the learning process. After some experimenting, however, it turned out that the Tiramisu model used could overcome the skewed class distribution without obvious problems so no class weighted loss function were introduced.

In the following subsection the data preprocessing steps are described. In the subsection after that the data augmentation strategies used during training are explained in more detail.

6.1 Preprocessing

When working with MRI-data in ANN-models there are two standard preprocessing procedures to be done before starting to use the data for training. The need for these procedures arises from the MRI technique itself and they are done in order to improve the quality of the data.

Firstly, MRI-scans are affected by so-called bias field distortion. This is a low-frequency signal, often present in older MRI-machines, which causes intensity variations across the same tissue in the scan. To alleviate this problem one typically uses a bias field correction algorithm. One of the most common of these algorithms is N4ITK [36], which is used for the MRI-scans in this work.

Secondly, MRI-scans have no fixed intensity scale. This means that the absolute intensity values between MRI-scans can vary even with the same patient and MRI-machine. This is

clearly not optimal for a model and some kind of normalization is needed. There are several ways to do this and the simplest would be to divide all intensities by the largest intensity of the scan, effectively setting an intensity span from 0 to 1. In this work however, a histogram equalization is performed instead. This not only solves the problem with varying intensity scales, it also uniforms the intensity distribution of the scan. Studying the intensity histogram before the histogram equalization (see Figure 6) one can see that the large majority of intensities span a relatively small range of values. By doing a histogram equalization one is spreading out the most frequent intensities and by doing so usually increasing the contrast of the scan. Figure 6 shows one MRI-scan slice from the BraTS dataset as well as the intensity histogram for that same scan, before and after the histogram equalization.

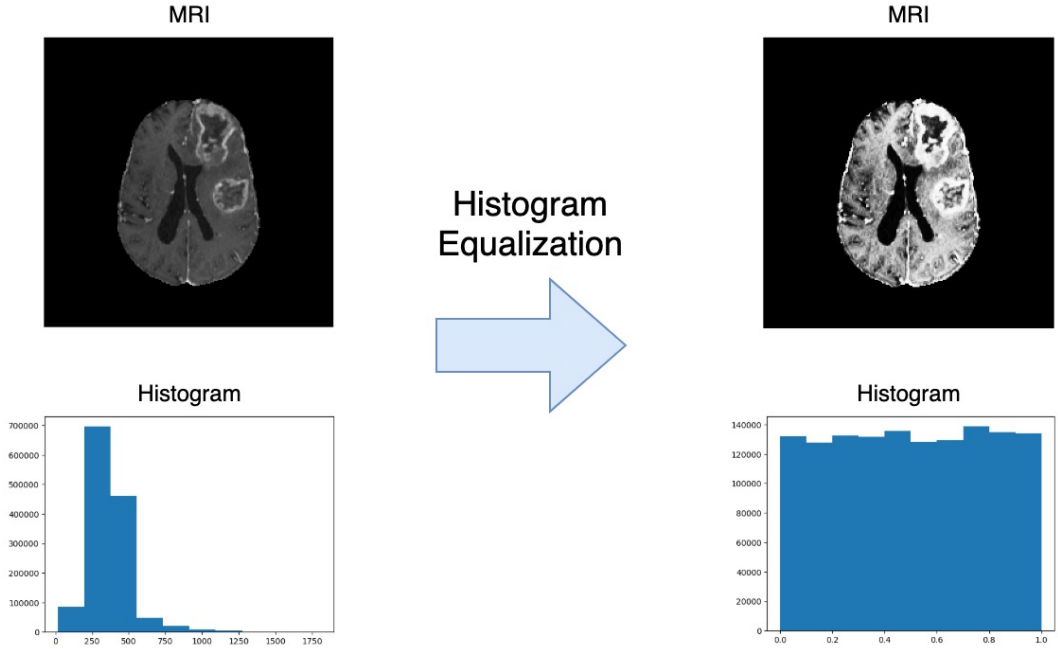
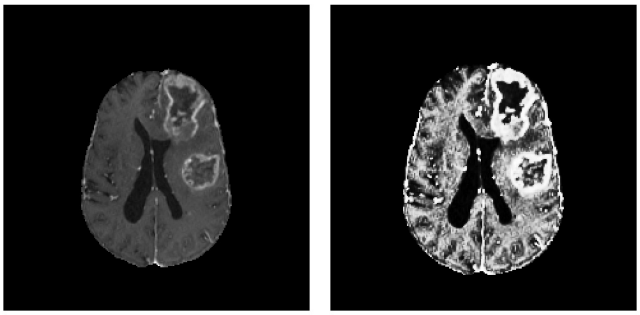


Figure 6: One slice of a MRI-scan and the corresponding histogram for the whole scan before and after histogram equalization.

In addition to bias field correction and histogram equalization the scans are also cropped as a part of the preprocessing. Since one MRI-scan is a 3D-structure but the Tiramisu network takes 2D-images as input each scan is split up in 2D-slices in the transverse plane (top to bottom). This means that the top and bottom zero-padding existing in the scans will only result in all-zero images and since that wouldn't provide the model with any information they are discarded. Each 2D-slice is also cropped from a size of 240×240 to 224×224 removing only zero-padding. This is done in order for the dimensions of the 2D-slices to work with the down- and up-samplings of the Tiramisu network as well as for saving computational memory during training. Naturally the ground truth segmentation maps are cropped in the same way as the scans. Figure 7 shows the same 2D-slice before and after preprocessing.



(a) **(b)**

Figure 7: (a) A slice of a MRI-scan before preprocessing. (b) A slice of a MRI-scan after preprocessing.

The 335 patient scans available for training are split up in a training set and a validation set, 80% training patients and 20% validation patients. A validation dataset is commonly used in machine learning to evaluate how models trained with different hyperparameters perform on unseen data and select the model that generalized the best during training, before finally evaluating the model's performance on the test set.

6.2 Data Augmentations

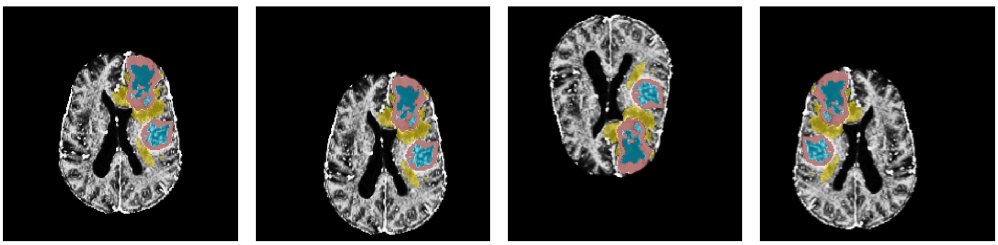
An important part of the training strategy described in Section 5.2 is the data augmentation. As Figure 4 shows, there are two different types of augmentation done, one weak and one strong.

6.2.1 Weak Augmentation

The weak augmentation is used on both labeled and unlabeled samples to make sure the network will not train on the exact same sample more than once, even if all the training samples are looped over multiple times. The augmentation methods used in the weak augmentation involves flipping and translation with different probability and magnitude. For each augmentation are all methods applied in random order. The complete details are presented in Table 4 and Figure 8 shows 3 examples of weakly augmented samples.

Table 4: The augmentation methods and parameter ranges used for the weak augmentation.

Method	Range
Flip Horizontally	50% probability
Flip Vertically	50% probability
Translation Horizontal	[-10%, 10%]
Translation Vertical	[-10%, 10%]



(a) Original **(b)** Weak augmentation **(c)** Weak augmentation **(d)** Weak augmentation

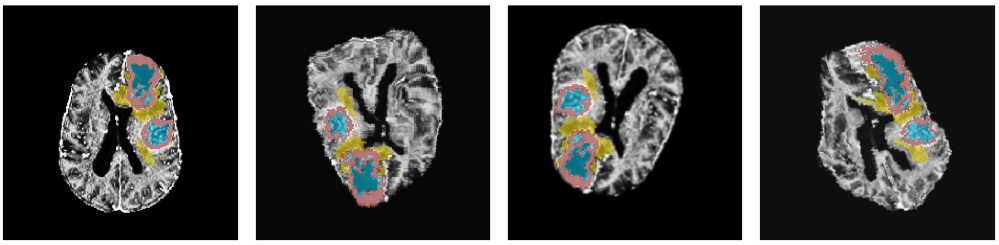
Figure 8: Three examples of weak augmentations along with the original MRI-slice.

6.2.2 Strong Augmentation

The purpose of the strong augmentation is, as explained in Section 5.2, to use in the EquiReg method with the unlabeled samples. As indicated, the strong augmentation is more extensive than the weak and was largely inspired from the augmentation strategy in [10]. Beyond the methods in the weak augmentation, the methods included in the strong augmentation are rotation, shearing, scaling and elastic distortion and for each augmentation are at least 3 – 8 of the methods applied in random order. Table 5 presents the methods and ranges for the strong augmentation and Figure 9 shows 3 examples of strongly augmented samples.

Table 5: The augmentation methods and parameter ranges used for the strong augmentation.

Method	Range
Flip Horizontally	50% probability
Flip Vertically	50% probability
Translation Horizontal	[-10%, 10%]
Translation Vertical	[-10%, 10%]
Rotation	[-20°, 20°]
Shear	[-20%, 20%]
Scale	[-10%, 10%]
Elastic Distortion	$\alpha = 720$, $\sigma = 24$



(a) Original **(b)** Strong augmentation **(c)** Strong augmentation **(d)** Strong augmentation

Figure 9: Three examples of strong augmentations along with the original MRI-slice.

As can be seen in Figure 8 and 9 the corresponding segmentation map augmented in the same way as the MRI-slice. Any new pixels created by the augmentation, e.g. when rotating or translating, will get the value 0 both when augmenting MRI-slices and segmentation maps. This is quite natural because a 0 pixel value corresponds to background in both cases.

All augmentations are done on the fly during the training process meaning there are no augmentations done as part of the preprocessing of data and the augmentations done in a training process are unique to that training process. The advantage of this is that there is no need to produce and store augmented data as part of the preprocessing. To be able to reproduce an identical augmentation later in training process, as needed in the EquiReg method (see Section 5.2), can the augmentations be generated using seeds. The augmentations were implemented using the `imgaug`-library [18].

7 Experiments

The experiments in this work aims to compare the purposed Equivariance Regularization (EquiReg) method, explained in Section 5.2, to ordinary supervised training, i.e., only the supervised part of the EquiReg-method. As a first step, a baseline is set in the form of models trained supervised on different percentages of the training dataset. After that, models are trained using the EquiReg-method with same labeled data as for the baseline models but also using all of the training data as unlabeled data. Comparing baseline models and EquiReg-models trained on the same labeled data would then indicate if utilizing available unlabeled data improves the models performance.

During a training session the model is regularly evaluated on the validation set. The model version with the lowest categorical cross-entropy loss on the validation set is considered the best.

The performance of the models is primarily measured with the Dice score of the three binary problems of segmenting the whole tumor, the tumor core and the enhancing tumor (see Section 6). In order to also get a convenient single performance value for each model a collective Dice score, the mean of the three Dice scores, will also be presented. These Dice scores are all calculated with respect to the ground truth, however, for the purpose of seeing if the EquiReg-models actually learns something that the baseline models does not is an equivariance Dice score also introduced. Instead of comparing a models predicted segmentation map and the ground truth the equivariance Dice score is the Dice score of two models predicted segmentation maps from the same input. One comes from letting the model segment the input and then strongly augmenting the segmentation map. The other comes from first strongly augmenting the input and then letting the model segment the augmented input. As the naming suggests, this means that the equivariance Dice score will reflect how equivariant the model is. Figure 10 illustrates the difference between the two Dice scores.

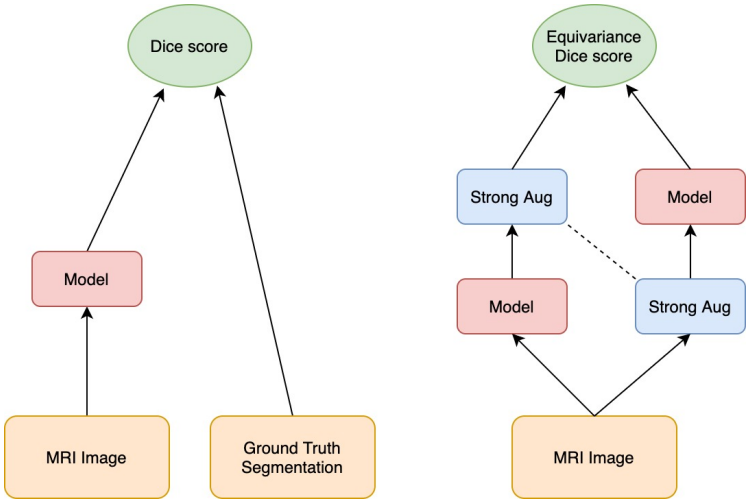


Figure 10: An illustration of the difference between the regular Dice score and the equivariance Dice Score.

The training strategy and the models are implemented in Tensorflow[2] 2.0.0 and trained using NVIDA TITAN RTX and NVIDIA GeForce RTX 2080 Ti GPUs. Code is available on Github[3].

7.1 Baseline models

The labeled training data percentages used to train the baseline models is 1%, 5%, 10% and 100% which corresponds to 3, 14, 27 and 268 patient MRI-scans. Naturally, different combinations of patients will contain different information. Hence, when training models on only a small percentage of patients it can be quite crucial which percentage of patients it is. With this in consideration 3 different seeds are used for shuffling the patients before selecting the first 1%, 5% and 10%. Moreover, since training deep learning models is not a deterministic process due to random weight initialization, data shuffling, etc., 3 models are trained with identical hyperparameters. Since no patient selection is needed when using all the labeled data, this results in 30 trained baseline models.

7.2 EquiReg-models

The EquiReg-models is trained with the exact same 1%, 5% and 10% labeled data as the baseline models, but also using all of the training dataset as unlabeled data. Specifically this

[2] https://www.tensorflow.org/
[3] https://github.com/gustavscholin/Thesis-EquiReg

means that each training batch given to the model contains both a number of labeled MRI-slices and a number of unlabeled MRI-slices. Furthermore, to investigate if the EquiReg-method would have any impact if all the training data were used supervised models are also trained using the EquiReg-method with 100% of the labeled data. Like for the baseline models, 3 models were trained for each set of hyperparameters resulting in 30 EquiReg-models.

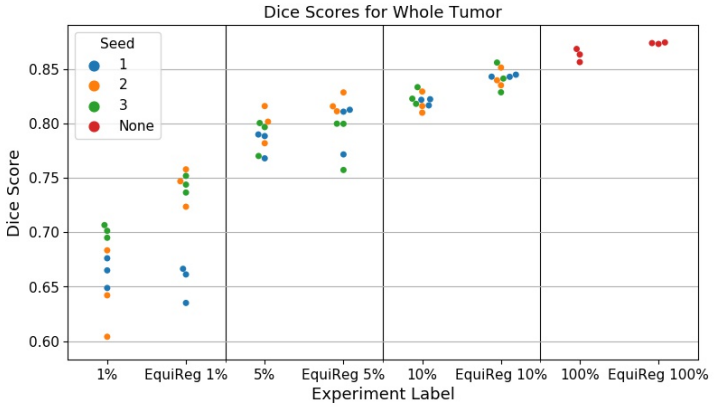
7.3 Results

In this section the results from the experiments are presented as quantitative and qualitative results. The quantitative results consist of different metric plots calculated from the models' regular and equivariance Dice scores, while the qualitative results are comparisons of segmentation maps from the different models.

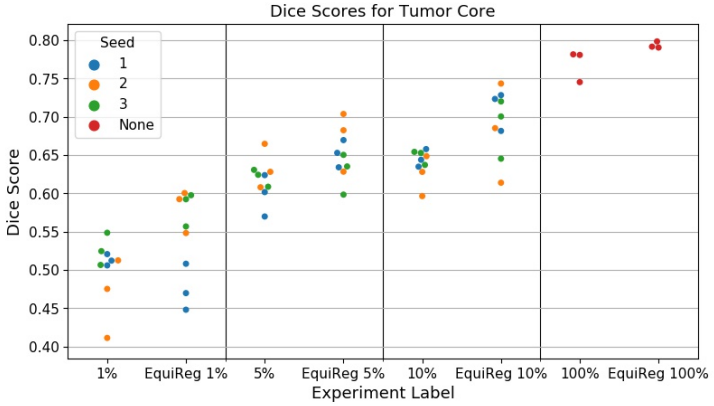
7.3.1 Quantitative Results

All quantitative results comes from evaluation on the test dataset.

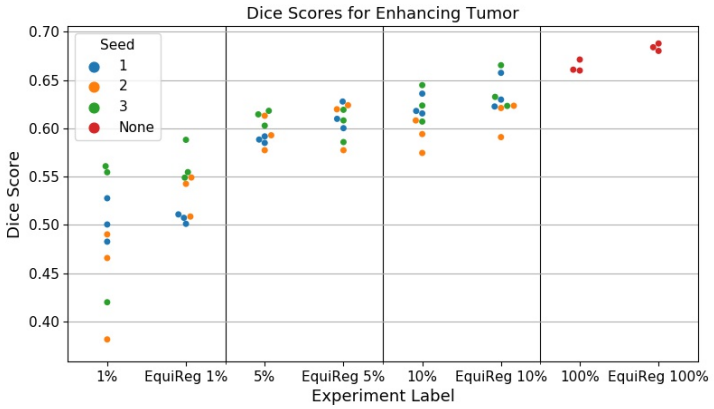
Figure 11 shows the Dice scores for all three sub-problems of each model and Figure 12 shows the collective Dice score for each model.



(a) Dice scores for segmentation of the whole tumor.



(b) Dice scores for segmentation of the tumor core.



(c) Dice scores for segmentation of the enhancing tumor.

Figure 11: Scatter plots showing all models' Dice scores of all three sub-problems. The color coding arises from which seed used in the labeled data selection. This means that same colored models with the same percentage have been trained on the same labeled training data. Notice that the Dice scores for the EquiReg-models are generally higher than the baseline models trained with the same amount of labeled data.

Figure 13 shows the mean Dice scores for the three sub-problems of all models with the same amount of labeled data and with the same method. Figure 14 shows corresponding plot for the collective Dice score. This highlights the performance difference between the EquiReg-method and the baseline at each percentage of labeled training data. The error for each point in Figure 13 and 14 is a 95% bootstrap confidence interval calculated from the raw Dice scores in Figure 11 and 12.

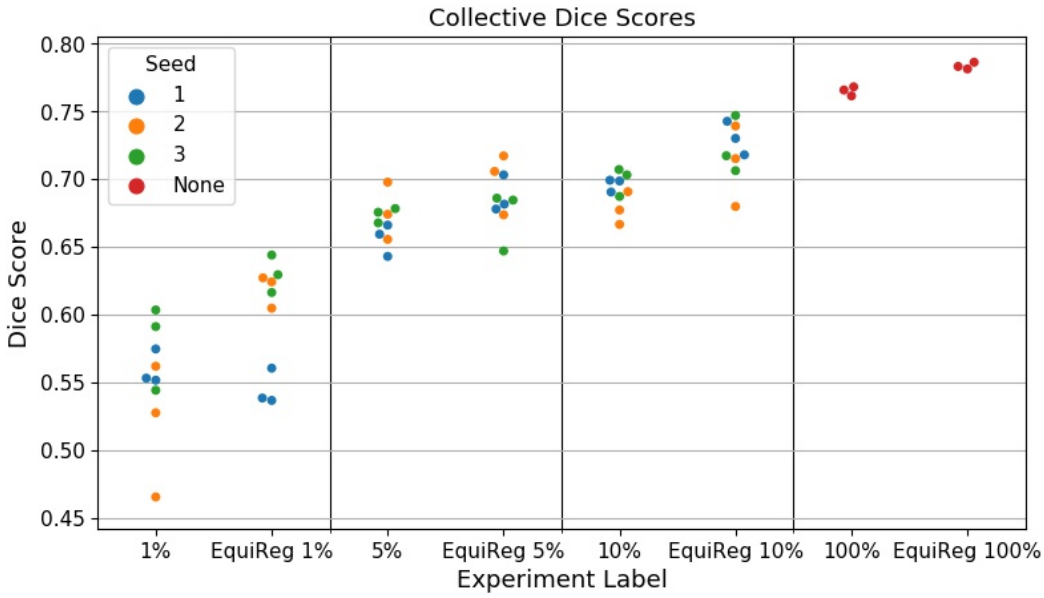
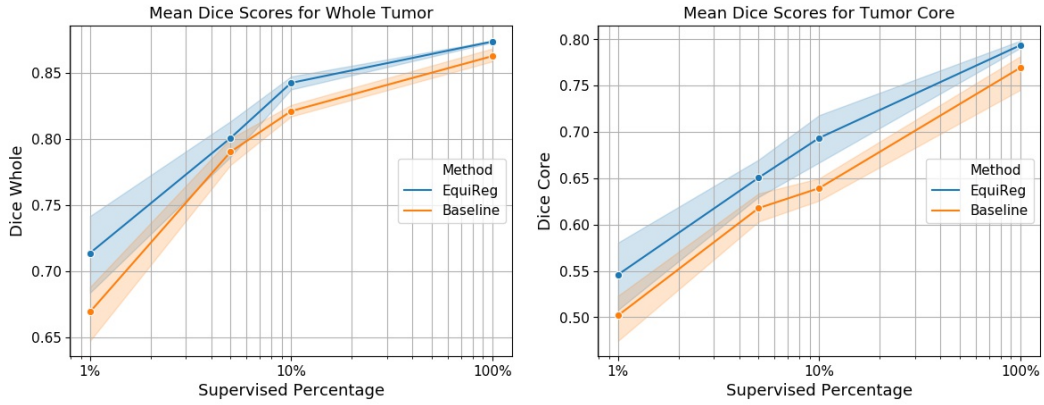
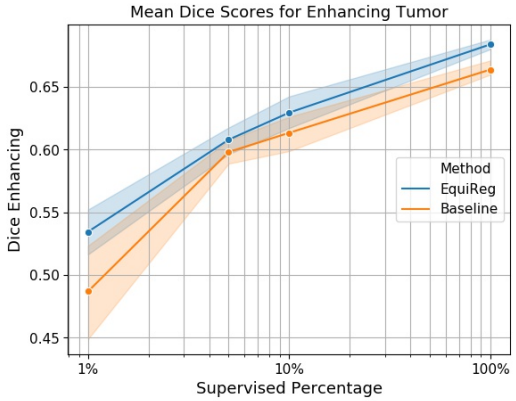


Figure 12: Scatter plot showing the collective Dice scores of all models. The color coding arises from which seed used in the labeled data selection. This means that same colored models with the same percentage have been trained on the same labeled training data. Notice that the Dice scores for the EquiReg-models are generally higher than the baseline models trained with the same amount of labeled data.



(a) Mean Dice scores for segmentation of the whole tumor.

(b) Mean Dice scores for segmentation of the tumor core.



(c) Mean Dice scores for segmentation of the enhancing tumor.

Figure 13: Line plots showing the performance in the three sub-problems for the EquiReg-method and the baseline method with different percentages of labeled data. One data point in the plots are the mean Dice scores of models trained with the same amount of labeled data and with the same method.

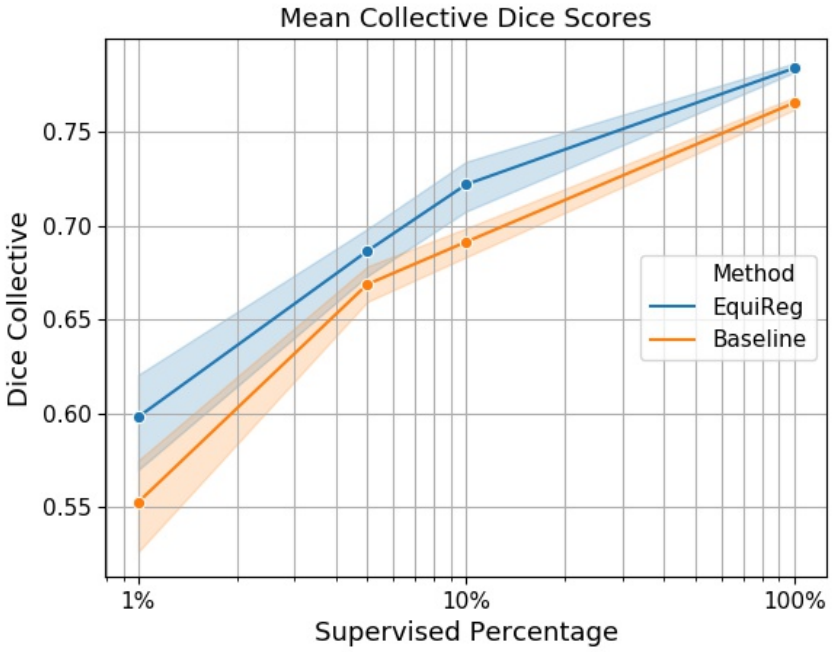
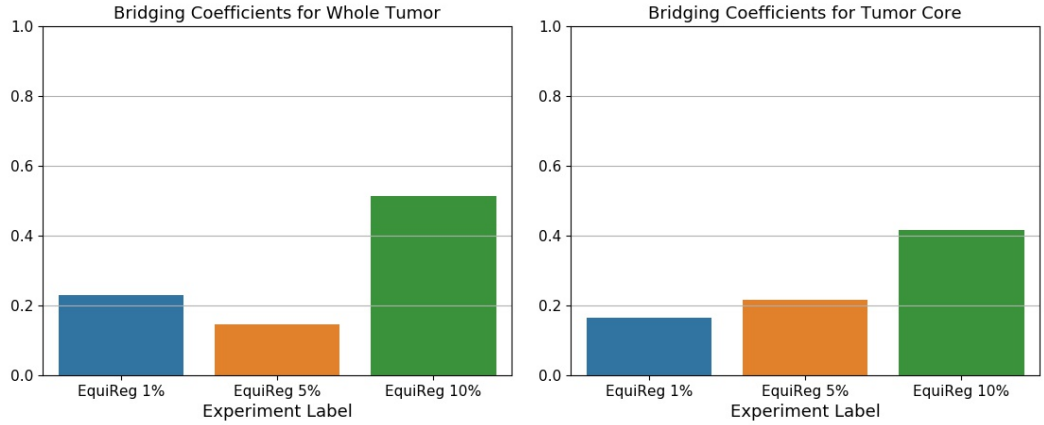


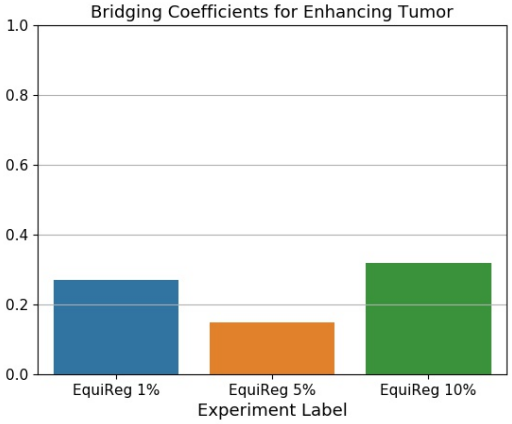
Figure 14: Line plot showing the collective Dice performance of the EquiReg-method and the baseline method with different percentages of labeled data. One data point in the plots are the mean Dice scores of models trained with the same amount of labeled data and with the same method.

Naturally there is a gap in Dice score performance between the baseline models using a small percentage of the labeled training data and the baseline models using all of the labeled training data. To measure how much of this gap the corresponding EquiReg-models can bridge a bridging coefficient is introduced. The bridging coefficients are calculated for the 1%, 5% and 10% EquiReg-models using the mean values in Figure 13 and 14. To clarify, a bridging coefficient of 0 corresponds to the performance of the small-percentage baseline while 1 corresponds to the performance of the baseline using all labeled training data. Figure 15 shows the bridging coefficients for all three sub-problems and Figure 16 shows the collective bridging coefficients.



(a) Bridging coefficients for segmentation of the whole tumor.

(b) Bridging coefficients for segmentation of the tumor core.



(c) Bridging coefficients for segmentation of the enhancing tumor.

Figure 15: Bar plots that show the 1%, 5% and 10% EquiReg-models' bridging coefficients for the three sub-problems. That is, how much of the baseline performance gap the EquiReg-method is bridging, the baseline performance gap being the performance difference of using all labeled data and a small percentage of labeled data in supervised training.

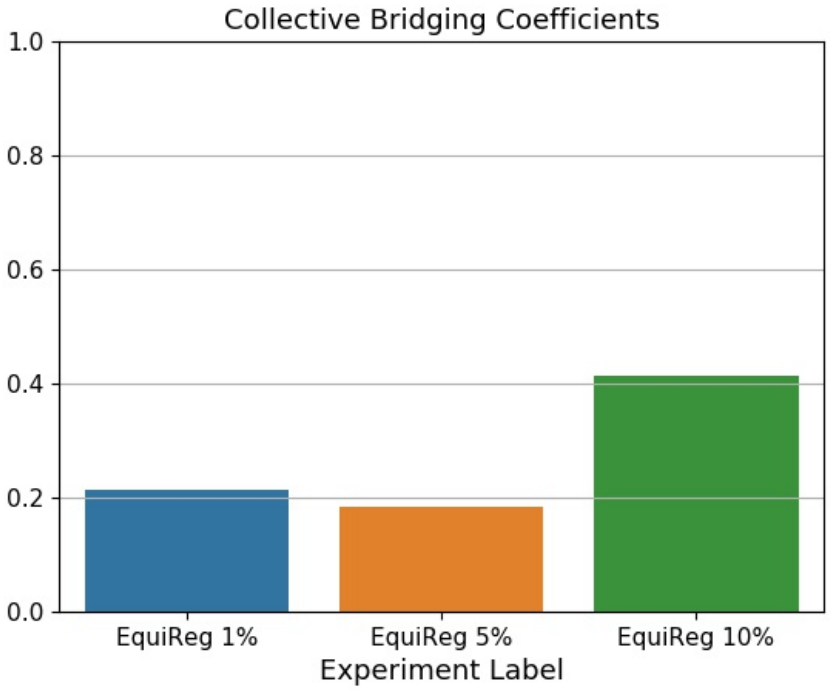


Figure 16: Bar plot that show the collective bridging coefficients for the 1%, 5% and 10% EquiReg-models. That is, how much of the baseline performance gap the EquiReg-method is bridging, the baseline performance gap being the performance difference of using all labeled data and a small percentage of labeled data in supervised training.

Figure 17 shows the collective equivariance Dice scores averaged over all models trained with the same amount of labeled data and with the same method. Also here are the error bars 95% bootstrap confidence intervals calculated from the models' raw equivariance Dice scores.

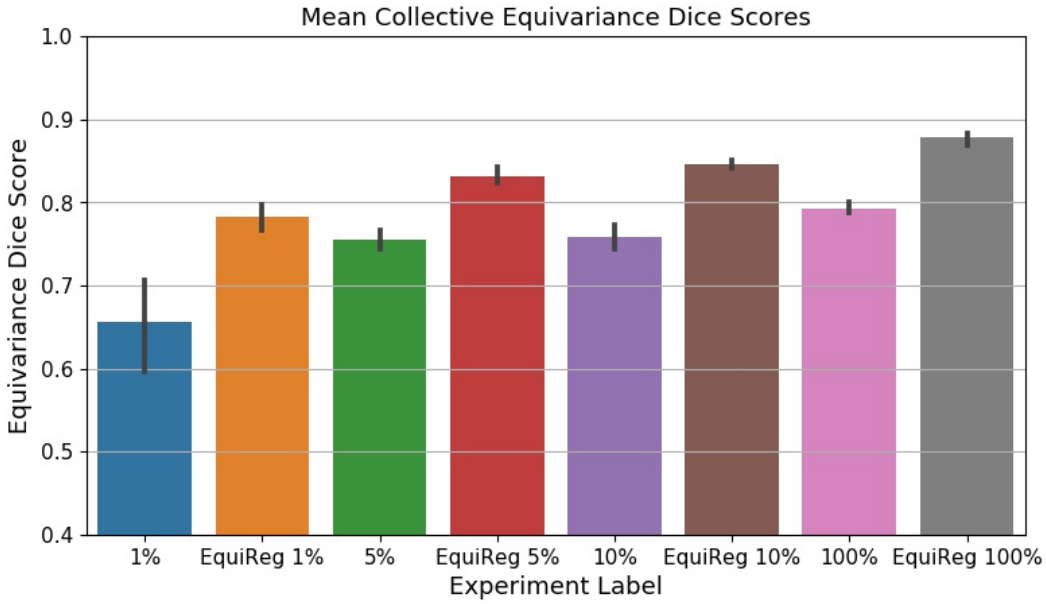
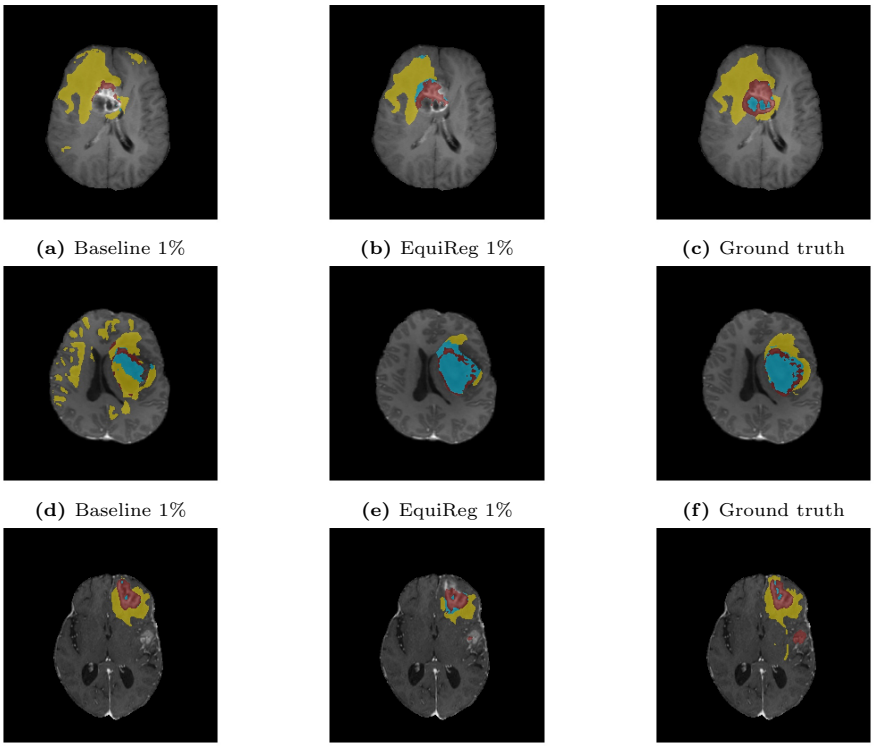


Figure 17: Bar plot showing the mean collective equivariance Dice scores.

7.3.2 Qualitative Results

Figure 18 – 21 shows examples of segmentation maps from baseline models and EquiReg-models compared with the corresponding ground truth. The models used are the best

comparable models, meaning the best performing models trained on the same labeled data. Since the ground truth segmentation maps for the test set are not publicly available these examples comes from the validation set.

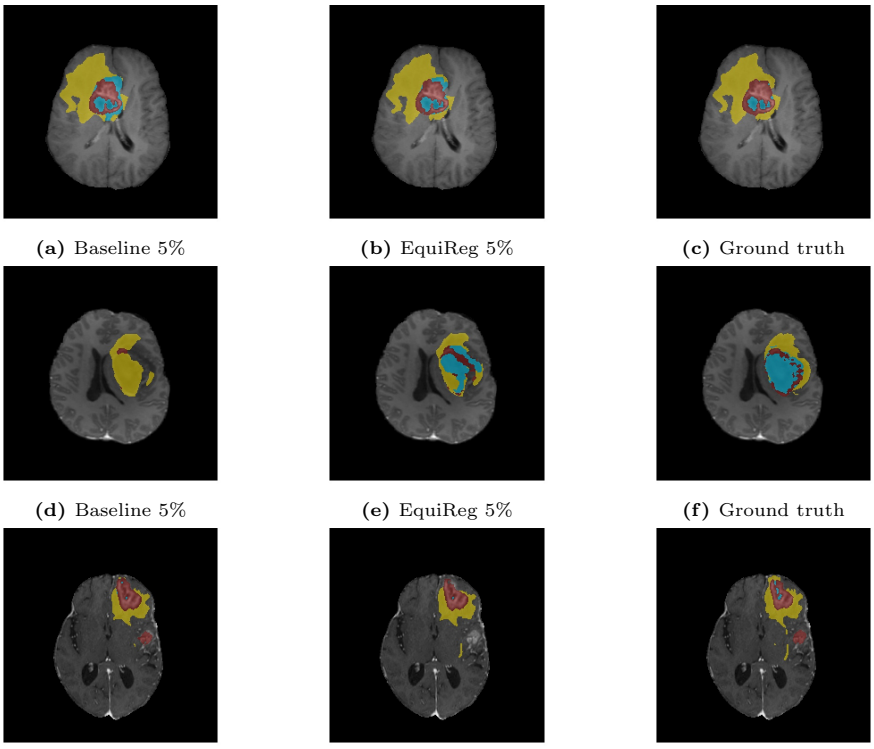


(a) Baseline 1% **(b)** EquiReg 1% **(c)** Ground truth

(d) Baseline 1% **(e)** EquiReg 1% **(f)** Ground truth

(g) Baseline 1% **(h)** EquiReg 1% **(i)** Ground truth

Figure 18: Segmentation examples on one MRI-slice from three different patients compared with the ground truth. The segmentation maps are produced by the best baseline and EquiReg-model trained on the same 1% of the labeled data.

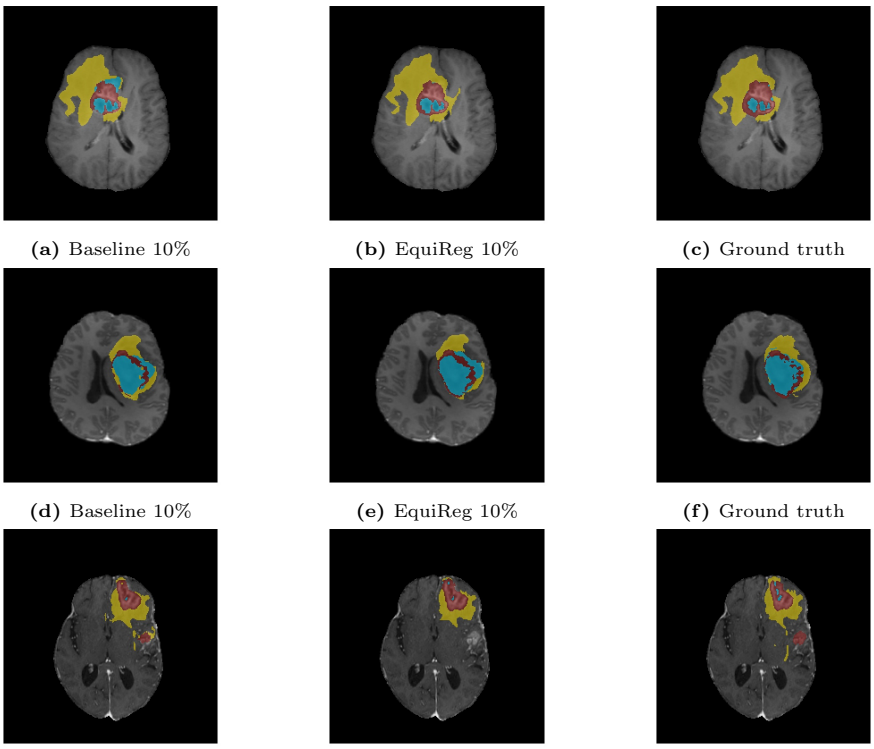


(a) Baseline 5% **(b)** EquiReg 5% **(c)** Ground truth

(d) Baseline 5% **(e)** EquiReg 5% **(f)** Ground truth

(g) Baseline 5% **(h)** EquiReg 5% **(i)** Ground truth

Figure 19: Segmentation examples on one MRI-slice from three different patients compared with the ground truth. The segmentation maps are produced by the best baseline and EquiReg-model trained on the same 5% of the labeled data.

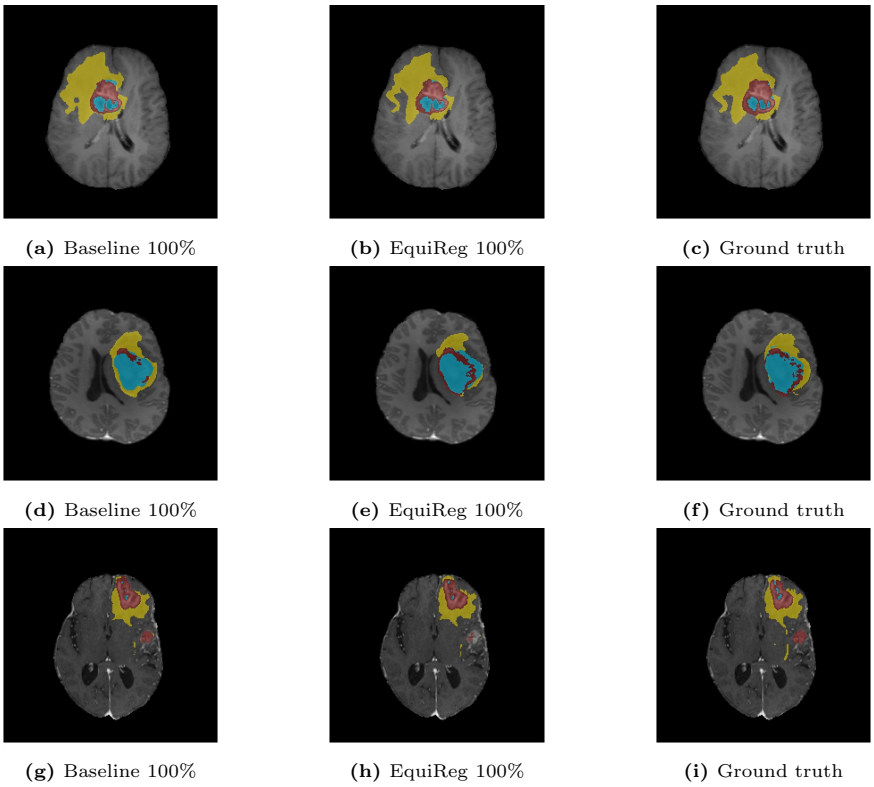


(a) Baseline 10% **(b)** EquiReg 10% **(c)** Ground truth

(d) Baseline 10% **(e)** EquiReg 10% **(f)** Ground truth

(g) Baseline 10% **(h)** EquiReg 10% **(i)** Ground truth

Figure 20: Segmentation examples on one MRI-slice from three different patients compared with the ground truth. The segmentation maps are produced by the best baseline and EquiReg-model trained on the same 10% of the labeled data.



(a) Baseline 100% **(b)** EquiReg 100% **(c)** Ground truth

(d) Baseline 100% **(e)** EquiReg 100% **(f)** Ground truth

(g) Baseline 100% **(h)** EquiReg 100% **(i)** Ground truth

Figure 21: Segmentation examples on one MRI-slice from three different patients compared with the ground truth. The segmentation maps are produced by the best baseline and EquiReg-model trained on all of the labeled training data.

7.4 Training Details

All baseline models are trained with a batch size of 4 labeled MRI-slices. The EquiReg-models are trained with batch size of 8 slices where 2 slices are labeled and 6 slices are unlabeled and the loss weighting factor λ is set to 1 (see Section 5.2). All models are trained with early stopping, meaning if the validation loss has not declined in a certain amount of training steps the training is stopped and considered completed. The gradient decent optimizer used during training is ADAM with a learning rate of 0.001.

8 Discussion

A quick glance at the results hints about a general increase in Dice score with the EquiReg-method compared to corresponding baseline. This indicates that using the EquiReg-method with unlabeled data actually helps the model to produce better segmentation maps. However, studying Figure 11 one can observe relatively high variance in Dice score within the same experiment label. Even models trained with the exact same hyperparameters can vary quite a lot, e.g., the 1% baseline with seed 2 and the 10% EquiReg with seed 2 in Figure 12. This makes it hard to draw any definitive conclusions about how much performance gain the EquiReg-method actually yields.

The relatively high variance in performance can be the result of several factors. First and foremost, recall that the training of the models is not a deterministic process and some variance is therefore expected. Moreover, it seems to be a tendency that less labeled data cause higher variance, which seems quite natural since small amounts of data can lead to more instability in the training process. Beyond that, some of the variance may also come from using the cross-entropy loss as supervised loss while measuring the performance using Dice score. This is because minimizing the cross-entropy loss correlates with, but is not equal to, maximizing the Dice score. Using a Dice score based supervised loss would most likely alleviate some of the variance problem, however, one distinct advantage with the

cross-entropy loss is that the gradient is simple and getting convergence during training is typically easier.

Another interesting thing to note in Figure 11 and 12 is that a model's performance in semi-supervised training is not only affected by how much labeled data is used but also which labeled data is used. Studying patient selection seed 1 for the 1%-models in Figure 11 and 12 one can see that there is no improvement for the EquiReg-method compared to the baseline but rather the opposite, while for the other two seeds the EquiReg-method performs better than the baseline. Similar tendencies can be seen for the 5%-models with patient selection seed 3. This suggests that the selection of labeled data is important for the EquiReg-method to work. However, visually studying the labeled data used to train the 1%-models (3 patients for each seed) gave no clear insight to what characterized a good selection from a bad one. It does, nonetheless, seems natural that these effects show in the 1%- and 5%-models where the labeled data selection intuitively should be more important than for the 10%-models.

In general, looking at the Dice scores in Figure 11 – 14 the feeling is that the performance increase of the EquiReg-method becomes clearer for 10% labeled data than for 5% and 1%. Looking at the bridge coefficients (defined in Sectioned 7.3.1) in Figure 15 and 16 it also shows that the EquiReg-method with 10% labeled data scores the highest, e.g., bridging 40% of the gap between the baseline and using all labeled data for the collective Dice score.

Looking at Figure 17 one can see that the mean collective equivariance Dice score is distinctively higher for the EquiReg-models than for their respective baselines. This is not too surprising since the unsupervised loss part in the EquiReg training strategy is effectively working to improve the equivariance of the model, but nevertheless the plot shows that the EquiReg-models learns something that the baseline models does not. This motivated to train the EquiReg-models that use all of the training data both supervised and unsupervised and as Figure 11 – 14 shows, these models actually performs better than the 100% baseline models. This indicates that a model can benefit from the unsupervised training on a data sample even if the very same data sample has already been exposed to the model during supervised training. Consequently, the use of the EquiReg-method is not necessarily restricted only to scenarios with small amounts of labeled data and large amounts of unlabeled data, but can possibly also be used to boost performance in training on large labeled data sets.

Even though the EquiReg-methods generally seems to perform a higher Dice score than their corresponding baseline it is important to note that this is not the case for all patients individually. The qualitative results in Figure 18 – 21 display two cases when the EquiReg-method performs better than the baseline, as well as one where the reverse is true. The first two examples (a – f) are from patients where the best EquiReg-model generally outperforms the best baseline model while the third example (g – i) is from a patient where the best baseline generally performs better.

9 Future Work

In future work it would be interesting to see if the performance improvement is consistent when using models other than the Tiramisu-model used in this work. Specifically in medical image segmentation the state-of-the-art FCNNs almost exclusively use 3D-convolutional layers, meaning that a MRI-scan can be processed by the FCNN without being split up in slices. This eliminates the information loss in the split-dimension and hence increases performance. 3D-models are, however, computationally and memory heavy to work with, which made them unfeasible for this work.

As was touched upon in the previous section one could also try a Dice score based loss to reduce the variance in performance between models. There are also other techniques for unsupervised training used in semi-supervised image classification that could be transferred to segmentation, e.g., in [33] where an approach with pseudo-labeling is used.

A possibly important part of this work where no extensive exploring is done is the data augmentations. Using many different augmentation methods means that there are a lot

of hyperparameters to be tuned in order to find an optimal schema for the task at hand; doing this manually would have been too time consuming for this work. However, there are methods to do this during training, as in [6] and [8], which could be implemented in this setting as well.

As stated in Section 8 the labeled data selection seems to be important for the EquiReg-method to perform well. Hence, the ability to distinguish a good labeled subset from a bad one would be desirable and more work can be devoted to finding heuristics for labeled subsets that work well with the EquiReg-method.

10 Conclusions

To conclude one can say that, despite that high variance in the results makes it hard to state definite numbers, the EquiReg-method seem to improve results compared to the supervised baseline. This includes, interestingly enough, the case where all training data is used which suggests that the method also can be used to boost supervised training on fully labeled datasets. The results indicates that more labeled data means clearer improvements and 10% labeled data seems like a good starting point for the method in a semi-supervised setting.

References

[1] Spyridon Bakas et al. "Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the BRATS challenge". In: *arXiv preprint arXiv:1811.02629* (2018).

[2] Spyridon Bakas et al. "Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features". In: *Scientific Data* 4 (2017), p. 170117.

[3] Spyridon Bakas et al. "Segmentation labels and radiomic features for the pre-operative scans of the TCGA-GBM collection". In: *The Cancer Imaging Archive* (2017).

[4] Spyridon Bakas et al. "Segmentation labels and radiomic features for the pre-operative scans of the TCGA-LGG collection". In: *The Cancer Imaging Archive* (2017).

[5] David Berthelot et al. "Mixmatch: A holistic approach to semi-supervised learning". In: *Advances in Neural Information Processing Systems*. 2019, pp. 5050–5060.

[6] David Berthelot et al. "ReMixMatch: Semi-Supervised Learning with Distribution Alignment and Augmentation Anchoring". In: *arXiv preprint arXiv:1911.09785* (2019).

[7] Özgün Çiçek et al. "3D U-Net: learning dense volumetric segmentation from sparse annotation". In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2016, pp. 424–432.

[8] Ekin D Cubuk et al. "Randaugment: Practical automated data augmentation with a reduced search space". In: *arXiv preprint arXiv:1909.13719* (2019).

[9] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[10] Hao Dong et al. "Automatic brain tumor detection and segmentation using u-net based fully convolutional networks". In: *annual conference on medical image understanding and analysis*. Springer. 2017, pp. 506–517.

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

[12] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[13] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[14] Gao Huang et al. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[15] Wei-Chih Hung et al. "Adversarial learning for semi-supervised semantic segmentation". In: *arXiv preprint arXiv:1802.07934* (2018).

[16] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[17] Simon Jégou et al. *The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation*. 2016. arXiv: 1611.09326 [cs.CV].

[18] Alexander B. Jung et al. *imgaug*. https://github.com/aleju/imgaug. Version 0.4.0. 2020.

[19] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[20] Durk P Kingma et al. "Semi-supervised learning with deep generative models". In: *Advances in neural information processing systems*. 2014, pp. 3581–3589.

[21] Samuli Laine and Timo Aila. "Temporal ensembling for semi-supervised learning". In: *arXiv preprint arXiv:1610.02242* (2016).

[22] 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*. 2015, pp. 3431–3440.

[23] Bjoern H Menze et al. "The multimodal brain tumor image segmentation benchmark (BRATS)". In: *IEEE transactions on medical imaging* 34.10 (2014), pp. 1993–2024.

[24] 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)*. IEEE. 2016, pp. 565–571.

[25] Sudhanshu Mittal, Maxim Tatarchenko, and Thomas Brox. "Semi-supervised semantic segmentation with high-and low-level consistency". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

[26] Andriy Myronenko. "3D MRI brain tumor segmentation using autoencoder regularization". In: *International MICCAI Brainlesion Workshop*. Springer. 2018, pp. 311–320.

[27] Openclipart. *Robot*. [Online; accessed April 4, 2020]. 2015. URL: https://openclipart.org/detail/226026/Robot.

[28] Ning Qian. "On the momentum term in gradient descent learning algorithms". In: *Neural networks* 12.1 (1999), pp. 145–151.

[29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *arXiv preprint arXiv:1505.04597* (2015).

[30] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[31] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. "Regularization with stochastic transformations and perturbations for deep semi-supervised learning". In: *Advances in neural information processing systems*. 2016, pp. 1163–1171.

[32] Shibani Santurkar et al. "How does batch normalization help optimization?" In: *Advances in Neural Information Processing Systems*. 2018, pp. 2483–2493.

[33] Kihyuk Sohn et al. "Fixmatch: Simplifying semi-supervised learning with consistency and confidence". In: *arXiv preprint arXiv:2001.07685* (2020).

[34] Nasim Souly, Concetto Spampinato, and Mubarak Shah. "Semi and weakly supervised semantic segmentation using generative adversarial network". In: *arXiv preprint arXiv:1703.09695* (2017).

[35] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[36] Nicholas J Tustison et al. “N4ITK: improved N3 bias correction”. In: *IEEE transactions on medical imaging* 29.6 (2010), pp. 1310–1320.

[37] Qizhe Xie et al. “Unsupervised data augmentation”. In: *arXiv preprint arXiv:1904.12848* (2019).

Master's Theses in Mathematical Sciences 2020:E18
ISSN 1404-6342

LUTFMA-3403-2020

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/